UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

JOSÉ MARTINS JUNIOR

**Modelagem e aplicação de regras comportamentais em ambientes
colaborativos, envolvendo agentes humanos e robóticos.**

São Carlos
2010

JOSÉ MARTINS JUNIOR

**Modelagem e aplicação de regras comportamentais em ambientes
colaborativos, envolvendo agentes humanos e robóticos.**

Tese apresentada à Escola de Engenharia
de São Carlos da Universidade de São
Paulo, como parte dos requisitos para
obtenção do título de Doutor em Engenharia
Mecânica.

Área de Concentração:
Dinâmica das Máquinas e Sistemas

Orientador:
Prof. Dr. Glauco Augusto de Paula Caurin

São Carlos
2010

AUTORIZO A REPRODUÇÃO E DIVULGAÇÃO TOTAL OU PARCIAL DESTE TRABALHO, POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

Ficha catalográfica preparada pela Seção de Tratamento
da Informação do Serviço de Biblioteca – EESC/USP

M386m	<p>Martins Junior, José</p> <p>Modelagem e aplicação de regras comportamentais em ambientes colaborativos, envolvendo agentes humanos e robóticos / José Martins Junior ; orientador Glauco Augusto de Paula Caurin. -- São Carlos, 2010.</p> <p>Tese (Doutorado-Programa de Pós-Graduação em Engenharia Mecânica e Área de Concentração em Dinâmica das Máquinas e Sistemas) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2010.</p> <p>1. Robótica. 2. Interação homem-máquina. 3. Ambiente colaborativo. 4. Linguagens formais. I. Título.</p>
-------	---

FOLHA DE JULGAMENTO

Candidato(a): Bacharel **JOSÉ MARTINS JÚNIOR**.

Tese defendida e julgada em 13/12/2010 perante a Comissão Julgadora:

Participação por videoconferência

Prof. Associado **GLAUCO AUGUSTO DE PAULA CAURIN** - (Orientador)
(Escola de Engenharia de São Carlos/USP)

marcelo Becker
Prof. Dr. **MARCELO BECKER**
(Escola de Engenharia de São Carlos/USP)

APROVADO

[Assinatura]
Prof. Dr. **LUIZ AUGUSTO MARTIN GONÇALVES**
(Escola de Engenharia de São Carlos/USP)

APROVADO

[Assinatura]
Prof^a. Dr^a. **JUNIA COUTINHO ANACLETO**
(Universidade Federal de São Carlos/UFSCar)

APROVADO

[Assinatura]
Prof. Dr. **LUIZ CAMOLESI JUNIOR**
(Universidade Estadual de Campinas/UNICAMP/campus de Limeira)

APROVADO

[Assinatura]
Prof. Associado **JONAS DE CARVALHO**
Coordenador do Programa de Pós-Graduação em
Engenharia Mecânica

[Assinatura]
Prof. Associado **PAULO CÉSAR LIMA SEGANTINE**
Presidente da Comissão da Pós-Graduação da EESC
Prof. Titular Denis Vinícius Coury
Suplente do Presidente da CPG-EESC-USP

À minha esposa Rita, meus filhos Thalles, Fernanda e Augusto, e aos meus pais José e Myltes, pelo carinho e pela confiança que depositaram em mim.

AGRADECIMENTOS

A Deus, por permitir minha presença aqui.

Aos meus pais, pelo amor e pelos ensinamentos que me deram e, principalmente, por servirem de modelos para a minha vida.

À minha esposa, com muito amor, pela imensa paciência e ponderação nos momentos mais difíceis desta caminhada.

Aos meus filhos, por me inspirarem a perseverança e por entenderem minha ausência em muitos momentos.

Ao meu orientador, pela fonte incessante de apoio e de motivação em minha pesquisa. Obrigado por me fazer acreditar que conseguiria.

Aos amigos Dalton Matsuo Tavares, Rafael Vidal Aroca, Bruno Jardim, Kleber de Oliveira Andrade e demais companheiros do grupo de pesquisa, pelas valiosas contribuições ao meu trabalho.

Aos amigos Marius Stücheli e Gisele Gonsalez Ito, pela sugestão e pela elaboração da fantasia de elefante.

Aos técnicos do laboratório José Francisco de Moraes, Sérgio Donizete Carvalho Ferreira e Jair Diego Antonietti, pelo excelente trabalho na confecção do tabuleiro e das peças, e pelo valioso apoio no reparo do robô.

“Ainda que eu falasse as línguas dos homens e dos anjos, e não tivesse amor, seria como o metal que soa ou como o sino que tine.”

1 Coríntios 13:1

RESUMO

MARTINS Jr, J. **Modelagem e aplicação de regras comportamentais em ambientes colaborativos, envolvendo agentes humanos e robóticos**. 2010. 156 p. Tese (Doutorado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2010.

Historicamente, o termo robô teve sua origem associada à forma humana e também ao seu comportamento. O ser humano desenvolveu, ao longo de seu processo evolutivo, capacidades mentais superiores como a memória, a linguagem, a vontade, entre outras. Tais faculdades permitem-lhe comparar informações obtidas do ambiente e de seus semelhantes com suas recordações e deliberar ações, ou realizar comunicações por meio de linguagens simbólicas, muitas vezes ambíguas. A área da cooperação robótica concentra estudos sobre a interação entre robôs e tem apresentado soluções para o controle adaptativo que, em sua maioria, dota os agentes robóticos de capacidades reativas a estímulos do ambiente. Porém, quando a interação envolve robô e humano, nota-se que as abordagens publicadas colocam apenas o ser humano no papel deliberativo. Essa solução mostra-se limitada, principalmente quando se busca por novas formas de interação que permitam um sistema robótico colaborar efetivamente com humanos e até tutorar o aprendizado destes. Com intuito de contribuir com a solução desse problema, é proposta e apresentada uma nova abordagem de controle, baseada em arquitetura distribuída e que permite a deliberação de comportamentos cooperativos e colaborativos. Além disso, o novo modelo de arquitetura permite operar multi-agentes distribuídos e, com isso, partes distintas de um robô manipulador. Para se validar a aplicabilidade do modelo, apresenta-se o sistema Scara3D, uma interface gráfica que representa o gerador do ambiente virtual, a camada mais baixa do contexto global da arquitetura. Os testes do ambiente virtual envolveram a tele-operação do robô, e seus resultados comprovam a integração e a comunicação entre os contextos global e local da arquitetura. Para que os agentes robóticos e humanos decidam e deliberem ações durante tarefas colaborativas, eles devem compartilhar um mesmo modelo mental dos elementos envolvidos nesse ambiente. A estratégia adotada consiste na representação por meio de uma linguagem simbólica, restrita e não-ambígua, capaz de ser compreendida por humanos e interpretada por computadores. Nesse sentido, as regras do ambiente colaborativo robô-humano são então definidas e descritas nos termos da L-Forum, uma linguagem para abstração de ambientes colaborativos. Um estudo de caso que envolve a colaboração robô-humano em um jogo da velha é descrito em detalhes, assim como o projeto e o desenvolvimento do *hardware* e do *software* que o operam. Os testes realizados descrevem situações que demandam do robô a seleção e a aplicação de regras diferentes, e seus resultados validam o processo de deliberação de comportamentos pelo sistema. Conclui-se, portanto, que o uso de regras colaborativas oferece um nível extra de abstração ao sistema, e o torna mais flexível e adaptável em ambientes compartilhados por seres humanos.

Palavras-chave: Robótica. Colaboração robô-humano. Regras comportamentais. Linguagens simbólicas.

ABSTRACT

MARTINS Jr, J. **Modeling and applying behavioral rules on collaborative environments, involving human and robotic agents**. 2010. 156 p. Thesis (PhD) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2010.

Historically, the term robot was originally associated to the human form and also to his behavior. The human being, during its evolutionary process, has developed high level mental abilities such as memory, language, will, among others. Such features allow him to compare information obtained from the environment and from his peers with their memories and deliberate actions or communications by means of a symbolic language, often ambiguous. The research area of robotics cooperation focuses on the interaction among robots and has presented solutions for the adaptive control that, in most cases, provides the robotic agents with reactive capabilities in response to environmental stimuli. However, when the interaction involves robot and human, the approaches that are available in the literature, assign the deliberative role only to the human. This is a constrained solution, especially when looking for new interaction forms that allow a robotic system to effectively collaborate with humans and to tutor the learning of them. Aiming to contribute to the solution of this problem is proposed and presented a new control approach based on distributed architecture that allows the deliberation of cooperative and collaborative behavior. Moreover, the new architecture model allows the interoperation of distributed multi-agents, and thus, distinct parts of a robot manipulator. The presented Scara3D system is used to validate the applicability of the model. The Scara3D is a graphical interface that represents the generator of the virtual environment, the bottom most layer of the global context of architecture. The tests of virtual environment involved the robot teleoperation, and its results prove the integration and communication between local and global contexts of architecture. For the robotic agents and humans decide and deliberate actions during collaborative tasks, they must share the same mental model of the elements involved in this environment. The adopted strategy consists of representation by means of a symbolic language, restricted and non-ambiguous, understandable by humans and interpretable by computers. In this sense, the rules of human-robot collaborative environment are then defined and described in terms of L-Forum, a language that allows abstracting collaborative environments. A case study involving human-robot cooperation in a tic-tac-toe is described in detail, as well as design and development of hardware and software that operate it. The tests describe situations that require the robot selection and application of different rules, and their results validate the process of behavior deliberation by the system. Therefore it is possible to conclude that the collaborative rules usage provides an extra abstraction level to the system, and makes it more flexible and adaptive in environments shared by humans.

Keywords: Robotics. Robot-human collaboration. Behavioral rules. Symbolic languages.

LISTA DE FIGURAS

Figura 1 – Modelo tradicional de controle aplicado a robôs móveis	28
Figura 2 – Modelo de controle baseado em comportamento	29
Figura 3 – Arquitetura híbrida deliberativa-reativa.....	31
Figura 4 – Modelo geral da projeção visual.....	38
Figura 5 – Modelo básico de um neurônio	46
Figura 6 – Hierarquia de Chomsky.....	50
Figura 7 – Representação esquemática da Máquina de Turing.....	52
Figura 8 – Programa representado por um grafo finito.....	53
Figura 9 – Foto do robô IBM SCARA do Laboratório de Mecatrônica – EESC/USP.....	82
Figura 10 – Foto da garra Kanguera	84
Figura 11 – Sistema de controle do robô SCARA, com plataforma aberta	85
Figura 12 – Decomposição de um robô SCARA em módulos.....	86
Figura 13 – Modelo multicamadas de controle cooperativo e colaborativo	87
Figura 14 – Arquitetura distribuída para o controle cooperativo e colaborativo.....	90
Figura 15 – Diagrama de Casos de Uso do Simulador Scara3D	92
Figura 16 – Diagrama de Componentes do Simulador Scara3D.....	96
Figura 17 – Principais classes do Simulador Scara3D.....	96
Figura 18 – Tela do Simulador Scara3D	98
Figura 19 – Imagens do robô Scara – a) webcam; b) foto	99
Figura 20 – Resultados da atuação do eixo 1 do robô	100
Figura 21 – Resultados da atuação do eixo 2 do robô	100
Figura 22 – Uma partida de “jogo da velha”	108
Figura 23 – Estampa do tabuleiro do “jogo da velha”	114
Figura 24 – <i>Reed switch</i> e seu encapsulamento.....	114
Figura 25 – Vista superior do tampo do tabuleiro.....	115
Figura 26 – Vista inferior do tabuleiro: placa, fonte e conexões.....	115
Figura 27 – Peças magnéticas confeccionadas para o jogo	116
Figura 28 – Esquema geral do sistema distribuído	116
Figura 29 – Foto do ambiente completo.....	117
Figura 30 – Robô vestido de elefante durante uma partida de “jogo da velha”	118
Figura 31 – Diagrama de classes do modelo conceitual do “jogo da velha”	121
Figura 32 – Visão geral das classes de regras do sistema	122
Figura 33 – Interface gráfica para ajuste do posicionamento do robô.....	124
Figura 34 – TCP do robô sobre a posição 15 do tabuleiro	124
Figura 35 – TCP do Scara3D sobre a posição 15 do tabuleiro	125
Figura 36 – Integração do Scara3D ao sistema	126
Figura 37 – Interface gráfica do Jogo da Velha.....	127
Figura 38 – Representação gráfica dos resultados dos testes de repetibilidade.....	132
Figura 39 – Seqüência de cenas gravadas em uma partida com o robô – 1	132
Figura 40 – Seqüência correspondente de telas do sistema – 1.....	133
Figura 41 – Seqüência de cenas gravadas em uma partida com o robô – 2	134
Figura 42 – Seqüência correspondente de telas do sistema – 2.....	135
Figura 43 – Seqüência de cenas gravadas em uma partida com o robô – 3	136
Figura 44 – Seqüência correspondente de telas do sistema – 3.....	137

LISTA DE QUADROS

Quadro 1 – Formalização da dimensão ator do M-Forum	75
Quadro 2 – Formalização da dimensão atividade do M-Forum.....	75
Quadro 3 – Formalização da dimensão objeto do M-Forum	76
Quadro 4 – Formalização da dimensão espaço do M-Forum.....	76
Quadro 5 – Formalização da dimensão tempo do M-Forum	77

LISTA DE TABELAS

Tabela 1 – Comparação entre M-Forum e outros modelos de regras.....	74
Tabela 2 – A sintaxe completa L-Forum.....	78
Tabela 3 – Exemplo de regra em L-Forum.....	80
Tabela 4 – Detalhamento do caso de uso “Interage com o robô”.....	92
Tabela 5 – Detalhamento do caso de uso “Aciona agentes locais”.....	93
Tabela 6 – Detalhamento do caso de uso “Inspeciona o ambiente virtual”	94
Tabela 7 – Especificações do robô IBM Scara 7545.....	95
Tabela 8 – Regras L-Forum para a passagem colaborativa de objetos	105
Tabela 9 – Conjunto de regras do “jogo da velha”	108
Tabela 10 – Regras do “jogo da velha” mapeadas em L-Forum	110
Tabela 11 – Descrição do protocolo de operação do <i>target</i>	123
Tabela 12 – Dimensões do código-fonte desenvolvido em Java.....	128
Tabela 13 – Mapeamento das posições antes do ajuste do PID.....	129
Tabela 14 – Outro mapeamento das posições antes do ajuste do PID.....	130
Tabela 15 – Mapeamento das posições depois do ajuste do PID.....	130
Tabela 16 – Testes de repetibilidade do link 3	131

SUMÁRIO

LISTA DE FIGURAS	8
LISTA DE QUADROS	9
LISTA DE TABELAS	10
1 INTRODUÇÃO	13
1.1. OBJETIVOS	15
1.2. ESTRUTURA DO DOCUMENTO	15
2 COOPERAÇÃO ENTRE ROBÔS	17
2.1. INTRODUÇÃO	18
2.2. SISTEMAS MULTI-ROBÔS COOPERATIVOS	20
2.2.1. Taxonomias para sistemas multi-robôs	24
2.3. COOPERAÇÃO E ROBÔS MODULARES	26
2.4. CONTROLE DISTRIBUÍDO APLICADO À ROBÓTICA	28
2.4.1. Controle baseado em comportamento	28
2.4.2. Controle distribuído baseado em agentes	30
2.4.3. Controle local <i>versus</i> controle global	31
2.4.4. Controle distribuído aplicado a robôs manipuladores	33
2.5. CONSIDERAÇÕES FINAIS	34
3 ATIVIDADES MENTAIS HUMANAS E A COMPUTAÇÃO	35
3.1. INTRODUÇÃO	35
3.2. A PERCEPÇÃO	36
3.2.1. O sistema visual	37
3.2.2. Visão computacional	43
3.3. A INTERPRETAÇÃO	44
3.3.1. O conexionismo	45
3.3.2. O simbolismo	46
3.3.2.1. Linguagem e significado	47
3.3.2.2. Linguagens formais	49
3.3.2.3. A Máquina de Turing	51
3.3.2.4. O Teste de Turing e a contestação de Searle: A sala chinesa	54
3.4. A CONSCIÊNCIA	55
3.5. A RELAÇÃO MENTE-CÉREBRO	56
3.5.1. O dualismo	56
3.5.2. O materialismo	57
3.5.3. O funcionalismo	58
3.6. TEORIA COMPUTACIONAL DA MENTE	59
3.6.1. Limitações da abordagem conexionista	61
3.6.2. Oportunidades para a abordagem simbólica	62
3.7. CONSIDERAÇÕES FINAIS	63
4 AMBIENTES COLABORATIVOS	65
4.1. INTRODUÇÃO	65
4.2. ELEMENTOS DE AMBIENTES COLABORATIVOS	66
4.2.1. Interação entre elementos em ambientes colaborativos	67
4.2.2. Linguagens e regras	68
4.3. COMPUTAÇÃO APLICADA A AMBIENTES COLABORATIVOS	69
4.3.1. Trabalho Cooperativo Apoiado por Computador	69
4.3.2. Agentes de <i>software</i>	70

4.3.3. Ontologias	72
4.3.4. Políticas de colaboração	73
4.3.4.1. O modelo M-Forum	75
4.3.4.2. A sintaxe L-Forum	77
4.4. CONSIDERAÇÕES FINAIS	80
5 COOPERAÇÃO ENTRE PARTES DE UM MANIPULADOR.....	81
5.1. ANÁLISE DO PROBLEMA	82
5.2. PROPOSTA DE UM NOVO MODELO DE ARQUITETURA	84
5.2.1. Arquitetura distribuída, baseada em multi-agentes.....	89
5.3. SCARA3D: SIMULADOR 3D PARA TESTE DA ARQUITETURA	91
5.3.1. Projeto e desenvolvimento do <i>software</i>	91
5.3.2. Testes e resultados	99
5.4. CONSIDERAÇÕES FINAIS.....	101
6 COLABORAÇÃO ROBÔ-HUMANO.....	103
6.1. PROPOSTA DO MODELO DE REGRAS COLABORATIVAS	104
6.1.1. Exemplo de aplicação do modelo de regras	105
6.2. <i>COLLABORATIVE TIC TAC TOE</i> : UM ESTUDO DE CASO	107
6.2.1. Análise: regras do “jogo da velha”	107
6.2.2. Definição do conjunto de regras colaborativas	109
6.2.3. Projeto e desenvolvimento do <i>hardware</i>	114
6.2.4. Projeto e desenvolvimento do <i>software</i>	118
6.2.4.1. Implementação do conjunto de regras colaborativas	122
6.2.4.2. Integração do robô IBM SCARA 7545 ao sistema.....	123
6.2.4.3. Integração do simulador Scara3D ao sistema	125
6.2.5. Testes e resultados	129
6.2.5.1. Testes de verificação: repetibilidade e precisão	129
6.2.5.2. Testes de validação: regras do jogo	132
6.3. CONSIDERAÇÕES FINAIS	138
7 CONCLUSÕES.....	141
7.1. SUGESTÕES PARA TRABALHOS FUTUROS	144
REFERÊNCIAS.....	147

1 INTRODUÇÃO

A Robótica é, sem dúvidas, uma das áreas de estudo mais intrigantes da atualidade. Além de demandar a interação em três grandes áreas do conhecimento – Mecânica, Eletrônica e Computação – ela possibilita o estudo e o desenvolvimento de soluções para problemas nas mais diversas áreas de aplicação. Entre essas áreas, destacam-se linhas de produção em indústria, cirurgia médica, exploração de cavernas e escombros, exploração em outros planetas, manipulação de lixo tóxico e de explosivos, assistência à visitação de espaços, próteses de membros do corpo humano, e brinquedos (GARCIA et al., 2007).

Aplicações industriais normalmente demandam a utilização de robôs manipuladores (CRAIG, 2004), ou braços robotizados, que se tornaram ótimas soluções para a execução de tarefas insalubres, perigosas ou repetitivas. Uma linha de montagem, típica da indústria automobilística, contém muitos exemplos dessas tarefas, como carregar/colocar/encaixar peças, aplicar solda e realizar pintura. Esse tipo de robô possui capacidades muito limitadas e dependentes de sua pré-programação, sendo assim, é capaz de realizar tarefas específicas com altos níveis de desempenho; por outro lado, está muito distante da antiga expectativa em relação aos robôs, como máquinas que operariam em conjunto com seres humanos, para estender as suas capacidades e os proteger.

A associação de robôs à figura humana é tema de muitas obras de ficção científica e inspirou clássicos, como Asimov (1978). O sonho antigo de compreender e simular o ser humano intriga especialistas e não especialistas há muitas gerações. Contudo, apesar da indústria cinematográfica há muito tempo explorar a livre interação entre robôs e seres humanos, essa realidade ainda está por ser atingida.

Trabalhos que descrevem a aplicação cooperativa de múltiplos robôs ganharam grande destaque na última década (PARKER, 2003). Modernas aplicações, que envolvem a cooperação de dois ou mais robôs, surgiram atualmente na indústria. Na busca por soluções adequadas para o controle distribuído, diversas arquiteturas foram propostas, muitas delas inspiradas em modelos de controle reativos baseados em comportamentos (BROOKS, 1986). Infelizmente, a maior parte dos trabalhos publicados descreve comportamentos coletivos simples, como os inspirados em insetos (CAZANGI; ZUGEN; FIGUEIREDO, 2005).

Outras estratégias, como a inicialmente proposta por Fischer, Müller e Pischel (1996), envolvem a aplicação de agentes híbridos em um modelo de controle deliberativo-reativo. Modelos desse tipo permitem que sejam acrescentadas à arquitetura de controle novas competências, baseadas em regras. Regras para ambientes colaborativos (CAMOLESI Jr; MARTINS, 2006) têm sido um dos assuntos de estudo em CSCW (*Computer Supported Cooperative Work* – Trabalho Cooperativo Apoiado por Computador). Tais ambientes prevêem a interação de agentes humanos que trabalham em um objetivo comum, como edição conjunta de documentos ou desenvolvimento de *software*.

Para Yanko e Drury (2004), a aplicação de técnicas e modelos de CSCW pode permitir o compartilhamento de uma concepção comum sobre uma tarefa realizada por robôs e humanos em um mesmo ambiente. Uma concepção comum a ambos os tipos de agentes pode ser definida em termos de uma linguagem restrita e não ambígua, que permita a sua interpretação por seres humanos e computadores. Pinker (2002) defende tal concepção ao argumentar que a Máquina de Turing realiza transformações simbólicas sobre uma linguagem interna, assim como a mente humana.

O tema sobre interação entre robôs e humanos pode ser encontrado em trabalhos publicados atualmente pela ciência. Entre eles, destacam-se aqueles que descrevem a interação física direta, seja para manipulação conjunta de um objeto (GROTEN et al., 2010), como para o treinamento pelo uso de interfaces hápticas (PARK; YOO; HOWARD, 2010). Em outros, a interação não prevê o contato físico, seja na predição de movimentos humanos (NAJMAEI; KERMANI, 2010) ou no aprendizado colaborativo por demonstração (PARK; HOWARD, 2010).

Porém, pode-se notar que a abordagem comum nos trabalhos apresentados reconhece o comportamento deliberativo (baseado em regras) apenas no ser humano, e atribui ao robô um papel secundário de copiar o comportamento humano ou reagir a estímulos do ambiente. Uma possível melhoria para esse modelo de colaboração consiste de dotar o agente robótico de capacidades deliberativas, ou seja, permitir que ambos os agentes, humano e robótico, compartilhem uma abstração semelhante do ambiente. A implantação de capacidades deliberativas em agentes robóticos permitiria também a evolução das formas de comunicação com humanos, e como consequência, a melhoria das formas de interação robô-humano.

1.1. OBJETIVOS

O presente trabalho tem, como principais objetivos, o estudo de situações que envolvem a colaboração entre agentes robóticos e humanos, bem como a concepção e a aplicação de um modelo de regras comportamentais para tais agentes.

Como objetivos específicos, apresentam-se: proposta de um modelo de arquitetura de sistema baseado em comportamentos de multi-agentes distribuídos, aplicado a partes de um robô manipulador; modelagem de um conjunto de regras comportamentais para um robô manipulador, aplicadas à colaboração com seres humanos; experimentação e teste do conjunto de regras em casos de colaboração.

1.2. ESTRUTURA DO DOCUMENTO

Este documento está organizado na forma apresentada a seguir. O capítulo 2 descreve a revisão da área de estudos da cooperação entre robôs e apresenta modelos de arquiteturas de controle propostos na literatura. No capítulo 3 é abordado o problema da representação da atividade mental humana e, em seguida, as diferentes abordagens computacionais para abstraí-la. No capítulo 4 descrevem-se os ambientes colaborativos, seus principais elementos, e os recursos da área de Computação que podem ser utilizados em suas representações. O capítulo 5 descreve a contribuição deste trabalho para a área de cooperação entre robôs e apresenta um novo modelo de arquitetura distribuída. No capítulo 6 descrevem-se as regras de colaboração entre robôs e humanos, que foram desenvolvidas neste trabalho. Além de conterem as principais contribuições deste trabalho, os capítulos 5 e 6 descrevem estudos de caso específicos, desenvolvidos e experimentados durante a elaboração do projeto, bem como os testes e seus resultados. As conclusões e sugestões para trabalhos futuros são apresentadas no capítulo 7.

2 COOPERAÇÃO ENTRE ROBÔS

O termo cooperar é definido por Houaiss (2001) na forma

cooperar, v. (1647 FSCout II 233) t.i.int. atuar, juntamente com outros, para um mesmo fim; contribuir com trabalho, esforços, auxílio; colaborar ... ▯ ETIM lat. *coopèror, aris, átus sum, ári* 'colaborar, trabalhar com outro(s)', v.dep. der. de *cum* (> co-) e *operari* 'trabalhar';

Interpreta-se, a partir da definição apresentada, que a cooperação envolve atuação conjunta (interação) de entidades distintas, com um objetivo comum. Se for considerado que a ação de trabalhar (laborar) envolve a realização de tarefas, mesmo inconscientemente (dado que consciência é uma faculdade ainda restrita a seres vivos), pode-se então tratar, como sinônimos, os termos operar e trabalhar. Portanto, consideram-se entidades aptas à cooperação e à colaboração os seres humanos, outros seres vivos e máquinas, capazes de realizar trabalho.

Apesar disso, neste documento, seguindo-se a prática dos trabalhos publicados na área, o termo cooperação é utilizado para descrever a interação entre máquinas. Por outro lado, assume-se que colaboração extrapola essa interação e inclui a participação de seres humanos no ambiente. Dessa forma, o objeto principal do estudo proposto – ambientes colaborativos – sugere a participação de seres humanos nos diversos cenários de aplicação, formas essas que serão discutidas nos próximos capítulos.

O estudo da cooperação entre robôs é especialmente importante neste trabalho, pois através dele podem-se estabelecer as relações entre os diferentes atores – sejam robôs ou partes deles – e o ambiente que compartilham. Tais relações devem também descrever os objetos e demais elementos que qualificam o ambiente cooperativo. Além disso, a estratégia de controle a ser adotada para a solução do problema deve permitir a operação de tais relações.

Dessa forma, este capítulo aborda o estudo da cooperação aplicada a sistemas robóticos. Inicialmente, é apresentada uma visão geral do assunto e, em seguida, a classificação dos diversos sistemas multi-robôs existentes. As demais seções do capítulo abordam formas de cooperação entre robôs modulares, além de estratégias para operação distribuída.

2.1. INTRODUÇÃO

Os robôs manipuladores há décadas têm sido utilizados pela indústria (CRAIG, 2004). É muito comum encontrar esse tipo de robô em linhas industriais de montagem, onde executam tarefas predefinidas. Toda a operação de cada robô é, portanto, previamente programada e consiste, em grande parte, de seu comportamento dentro de um espaço de trabalho previamente estabelecido.

No entanto, é crescente o uso de sensores nos robôs e também a sua diversidade (óticos, táteis, dentre outros), o que permite a modelagem do ambiente em resposta a mudanças nele detectadas e o ajuste do controle durante a execução das tarefas. Tal capacidade pode proporcionar robustez ao robô, permitindo-lhe adaptar-se a situações não previstas do ambiente. Tais situações permitem incluir a mudança da posição do objeto que será manipulado, bem como introdução de obstáculos em seu espaço de trabalho. Por outro lado, a aplicações que envolvem operação adaptativa pode degradar o desempenho do sistema, por exigir maior uso de computação dos dados captados pelos sensores. Vários trabalhos foram propostos na busca da solução de tal problema, mediante a aplicação de inteligência artificial e de visão computacional (ICARCV, 2004), e também novas abordagens para arquiteturas de controle (HERRERA et al., 2003).

Outra abordagem, que tem se tornado bastante comum em trabalhos da área, envolve a aplicação dos princípios de divisão e de distribuição de tarefas (GERKEY; MATARIĆ, 2004). A exemplo do que ocorre em sistemas computacionais distribuídos, um equipamento de grande porte, em diversos casos, pode ser substituído por vários equipamentos de menores porte e custo, que permitam o mesmo (ou melhor) desempenho geral. Para Cao, Fukunaga e Kahng (1997), alguns fatores influenciam a aplicação de vários robôs, sob um comportamento coletivo: a tarefa em questão é muito complexa para um único robô; a utilização de vários robôs pode ser mais prática, econômica, flexível ou tolerante a falhas; a abordagem em questão envolve estudo de interações entre robôs como, por exemplo, na imitação de grupos sociais ou de animais.

A inspiração em modelos de comportamento observados em seres vivos foi a base dos primeiros trabalhos na área da cibernética. Segundo Holland (2001), no final da década de 1940, o Dr. William Grey Walter, neurofisiologista inglês, desenvolveu algumas tartarugas biônicas, com um sistema simples de motores

alimentados por bateria que movimentavam três rodas¹. Sensores de luz e de contato foram também conectados ao circuito analógico com válvulas eletrônicas. Dessa forma, as tartarugas de Walter apresentaram vários tipos de comportamentos previsíveis, como seguir uma fonte luminosa e desviar de obstáculos após colisão.

Brooks (1991) apresentou a necessidade de se desenvolverem sistemas robóticos que agreguem capacidade de reação ao ambiente (mundo real), e justifica sua decepção com trabalhos fundamentados em simulação, ou definidos para a solução de problemas pontuais. Para Brooks, é importante que as relações entre robôs sejam baseadas em comportamento, de forma semelhante às ocorridas entre animais. O pesquisador ainda afirma que “O mundo é o seu próprio melhor modelo”, sendo assim, um sistema robótico que interage com o ambiente deve representar o modelo presente no mundo real, ao contrário de criar o seu próprio modelo interno, muitas vezes simplificado ou adaptado aos seus limites. O trabalho de Brooks serviu como base para a fundamentação de um grande número de pesquisas atuais, relacionadas à aplicação de modelos comportamentais biológicos para o controle e a operação de grupos de robôs homogêneos ou heterogêneos.

Dos comportamentos coletivos presentes na natureza, dois deles merecem especial atenção para o estudo em robótica (CAO; FUKUNAGA; KAHNG, 1997): o eusocial e o cooperativo. O eusocial pode ser encontrado em várias espécies de insetos, como formigas e abelhas, devido à sobreposição de gerações em um ninho, ao cuidado cooperativo com a prole e à especialização de tarefas. Cazangi, Zugen e Figueiredo (2005) apresentam um trabalho inspirado em tal comportamento.

O comportamento cooperativo pode ser notado em seres vivos mais complexos (como os vertebrados), inclusive entre os humanos; tal comportamento justifica-se pela motivação social, o que implica que indivíduos isolados devem sentir a necessidade ou o desejo pela cooperação. Parker (2003) esclarece que os principais trabalhos atualmente desenvolvidos nessa linha referem-se a modelos de comportamento predador-presa ou que envolvem competição, a exemplo de comportamentos reativos em robôs jogadores de futebol (KITANO et al., 1997). Cao, Fukunaga e Kahng (1997) afirmaram que um sistema de múltiplos robôs apresenta comportamento cooperativo se, ao ser realizada uma tarefa por algum mecanismo cooperativo, notar-se um incremento na utilidade total do sistema.

¹ Fonte: <http://www.ias.uwe.ac.uk/Robots/gwonline/gwonline.html>

2.2. SISTEMAS MULTI-ROBÔS COOPERATIVOS

A origem dos estudos na área de multi-robôs cooperativos surgiu da necessidade em se controlar grupos de robôs para executarem tarefas e para atingirem um objetivo comum. Muitas aplicações são compostas de tarefas inerentemente distribuídas, seja em relação ao espaço, ao tempo ou à funcionalidade (PARKER, 1998), o que pode justificar a divisão entre vários participantes. A distribuição de tarefas pode permitir ganhos no desempenho e na confiabilidade do sistema, pelo uso de mecanismos que permitam o paralelismo e a redundância. Pelo paralelismo entende-se que várias tarefas podem ser executadas simultaneamente, enquanto que a redundância pode ser atingida pela substituição de integrantes do grupo que se apresentem inoperantes.

O modelo real que inspirou tal comportamento relaciona-se à atribuição de tarefas a grupos de pessoas que podem, ou não, ter diferentes habilidades e desempenhos. Algumas aplicações são divididas em tarefas diferentes, que exigem papéis distintos dos integrantes do grupo. Essa forma de cooperação aplica-se aos sistemas multi-robôs heterogêneos.

A heterogeneidade entre os robôs pode se valer de aspectos de construção do *hardware*, ou mesmo do *software*. Pesquisas na área da computação reconfigurável (DEHON; WAWRZYNEK, 1999), têm permitido a adoção de estratégias para adaptação de perfis de *software* adequados ao comportamento esperado de cada agente robótico, de acordo com uma situação ou necessidade. Um dos grandes problemas que impedem o rápido avanço nesta área é a atual limitação de desempenho de processamento, o que torna a adaptabilidade pouco aplicável. Porém, o uso de dispositivos reconfiguráveis – a exemplo de FPGAs (*Field-Programmable Gate Arrays*) – em arquiteturas dinamicamente reprogramáveis tem impulsionado o avanço nessa área (COMPTON; HAUCK, 2002).

Cao, Fukunaga e Kahng (1997) reconhecem cinco linhas de pesquisa que compõem a concepção de mecanismos de cooperação. A primeira linha de pesquisa refere-se à arquitetura do grupo, que fornece a infraestrutura sobre a qual o comportamento coletivo é construído, e determina as capacidades e limitações do sistema. Diversas arquiteturas para sistemas cooperativos de multi-robôs foram propostas (PARKER, 1998; CHAIMOWICZ et al., 2001).

Uma arquitetura pode ser caracterizada sob os seguintes aspectos:

- Centralização/descentralização: uma arquitetura é centralizada se possuir um único agente de controle, ou descentralizada, se os agentes tiverem iguais papéis (distribuída) ou se dispostos em uma hierarquia de controle (hierárquica);
- Diferenciação: se o grupo é formado por robôs homogêneos ou heterogêneos;
- Estrutura de comunicação: três tipos distintos de comunicação são aplicáveis em arquiteturas para multi-robôs cooperativos, que envolvem, respectivamente, interação via: memória compartilhada, sensoramento e troca de mensagens;
- Modelagem dos outros agentes: aplica-se à representação das intenções, opiniões, ações, capacidades e estados dos outros agentes do grupo.

A segunda linha de pesquisa dedica-se à solução de conflitos sobre recursos que, de forma geral, referem-se ao espaço compartilhado, aos objetos manipuláveis ou ao meio de comunicação. Desses recursos, o que tem recebido maior atenção dos pesquisadores é o espaço compartilhado, na busca de soluções para problemas como controle de tráfego, planejamento de trajetória, proteção contra colisões e *deadlocks* (VIVALDINI et al., 2010a).

A terceira linha busca descrever a origem da cooperação no sistema, ou seja, que tipo de comportamento coletivo será utilizado, o eusocial ou o comportamento cooperativo. Como mencionados anteriormente, ambos os tipos são aplicados em diferentes arquiteturas.

A quarta linha de pesquisa refere-se aos problemas de aprendizado, que pode ser aplicado em diferentes níveis do controle (global ou local) do grupo de agentes. O aprendizado pode fornecer maior otimização no desempenho do grupo, além de permitir formas de implementação do controle adaptativo, o que viabiliza o conjunto suportar alterações dinâmicas no ambiente.

A última linha de pesquisa aplica-se ao estudo de problemas geométricos, que são inerentes e específicos ao tipo de aplicação ou problema abordado. Estratégias distintas têm sido propostas para problemas de planejamento de trajetória, formação e marcha, e geração de padrões.

Em Parker (2003) são apresentadas diversas linhas de pesquisas em sistemas multi-robôs. A autora descreve que os primeiros trabalhos com robôs

distribuídos começaram na década de 1980, e os primeiros assuntos tratados foram sistemas robóticos celulares (reconfiguráveis), planejamento de movimentação para multi-robôs e arquiteturas para cooperação entre multi-robôs.

Desde então, foi possível notar o avanço em, pelo menos, oito áreas de pesquisa bem definidas:

- Inspiração biológica: sistemas atuais, como o descrito em Halloy et al. (2007), valem-se do paradigma do controle baseado em comportamento (BROOKS, 1986). As aplicações mais comuns utilizam modelos simples de regras locais, para cada agente, mas que coletivamente apresentam comportamentos mais complexos, como as habilidades de agrupamento, perseguição de trilhas, agregação, dentre outras; tais modelos inspiram-se em animais pequenos, como formigas, abelhas e aves. O principal desafio nesta área consiste na aplicação de técnicas adequadas de aprendizado, e assim tornar possível a imitação de comportamentos coletivos de animais maiores, como as formas de cooperação e de competição entre humanos;
- Comunicação: o benefício da comunicação para certo número de tarefas já foi comprovado em diversos trabalhos científicos (PROROK; CIANCI; MARTINOLI, 2010) (CHAKRABORTY; SYCARA, 2010) (GIL et al., 2010). A comunicação pode ocorrer implícita ou explicitamente, sendo esta última o foco atual da pesquisa, em busca de se prover canais confiáveis e tolerantes a falhas para a troca de informação entre agentes, em tempo real (PEI; MUTKA; XI, 2010);
- Arquiteturas, planejamento de tarefas e controle: as pesquisas nesta área tratam problemas como a seleção de ações, a delegação de autoridade e de controle, a estrutura de comunicação, robôs homogêneos versus robôs heterogêneos, coerência em ações locais e resolução de conflitos. Tais esforços são dedicados, principalmente, à obtenção de capacidades, como planejamento de tarefas, tolerância a falhas, controle de *swarms* (enxames de robôs homogêneos) (MONDADA et al., 2005)(CORRELL; RUTISHAUSER; MARTINOLI, 2006) e planejamento humano de missões. O grande desafio nesta área, para os próximos anos, é tentar integrar as várias arquiteturas propostas por diversos grupos de pesquisa;

- Localização, mapeamento e exploração: a maioria dos trabalhos desenvolvidos nesta área (VIVALDINI et al., 2010b) (ISER; MARTENS; WAHL, 2010) limita-se a testar e adaptar algoritmos existentes para operação de um robô, estendendo-os para a operação de múltiplos robôs;
- Transporte de objetos e manipulação: esta área de pesquisa define um tipo de aplicação já tradicional em trabalhos com multi-robôs. A tarefa mais comum é a que envolve a aplicação de um grupo de agentes homogêneos ou heterogêneos para empurrar ou carregar um objeto. O desafio aqui é o estudo do transporte para terrenos irregulares;
- Coordenação de movimentos: trata-se de um tópico tradicional da área, e envolve temas como o planejamento de trajetória para multi-robôs, controle de tráfego, geração e manutenção de formação. Novos comportamentos têm sido abordados nesta área, como a localização e a perseguição de alvos. Outra evolução é o espaço de trabalho, tradicionalmente 2D, e atualmente passam a considerar ambientes 3D. Infelizmente, a limitação aqui continua no poder de processamento embarcado, para permitir a computação em tempo real da adaptação dinâmica da movimentação dos robôs;
- Robótica reconfigurável: apesar de ter inspirado os primeiros trabalhos em robótica distribuída, pouco avanço significativo foi notado nesta área, nos últimos anos. Os principais destaques ficam para robôs modulares que possuem a capacidade de alterar o seu formato físico, conectando e desconectando módulos. Os trabalhos baseiam-se em capacidades teóricas desse tipo de robôs, quanto à robustez, versatilidade e auto-reparo. As principais aplicações demonstradas nessa área limitam-se à auto-reconfiguração adaptativa para navegação em diferentes tipos de ambientes, como chão liso, escadas, e terreno irregular. Os módulos podem se reconfigurar em formas que lembram uma cobra, uma aranha ou uma esteira rolante. O desafio para a área é demonstrar ganho de desempenho em aplicações realmente úteis e práticas;

- Aprendizado: muito se tem publicado sobre o aprendizado de multi-agentes, mas as pesquisas com o aprendizado de multi-robôs ainda são escassas. As principais aplicações para a pesquisa em aprendizado são as que envolvem modelos comportamentais baseados em predador-presa, tarefas de traslado de caixas (objetos), futebol de robôs (FARIA et al, 2006) e observação cooperativa de alvos. O principal desafio nesta área é tratar formas de aprendizado para tarefas inerentemente cooperativas, que não permitem ser decompostas em subtarefas independentes, para serem alocadas aos agentes pelos métodos tradicionais.

Outra questão colocada por Parker (2003) é “Como será possível aos seres humanos controlar facilmente times de multi-robôs?”. Nesse contexto, a autora assinala que pouco se tem feito para melhorar a relação entre seres humanos e robôs. Além disso, fica claro que o comportamento cooperativo, presente entre animais maiores, tem sido apenas tomado como inspiração para o desenvolvimento de modelos cooperativos em times de multi-robôs. Muito pouco se tem observado da exploração do comportamento cooperativo, envolvendo times mistos de robôs e de seres humanos.

2.2.1. Taxonomias para sistemas multi-robôs

Sistemas multi-robôs têm sido propostos e implementados nas últimas décadas, seguindo as mais variadas conformações e para as mais diversas aplicações. Pode-se notar, porém, que a ausência de padronização torna mais difícil o projeto de tais sistemas para a solução de algum problema. Uma das decisões que o projeto de um sistema robótico deve considerar é se o uso de vários robôs é mais vantajoso que um único robô para a tarefa em questão.

Apesar de explicitamente desconsiderada como uma linha de pesquisa por Cao, Fukunaga e Kahng (1997), a decomposição e a alocação de tarefas tiveram um grande destaque em trabalhos científicos, inclusive na definição de taxonomia baseada em alocação de tarefas (GERKEY; MATARIĆ, 2004). Dudek, Jenkin e Milios (2002) apresentam uma abordagem da análise do impacto do tipo da tarefa sobre a organização de times de robôs, classificando tarefas que:

- Requerem o uso de múltiplos agentes: basicamente, aquelas que se apresentam em subtarefas separadas em relação ao espaço e que requerem certo nível de sincronismo;
- São tradicionalmente multi-agentes: descrevem tarefas tipicamente paralelas, que exigem um pouco de comunicação ao início ou ao final de sua execução;
- São normalmente aplicadas a um único agente: tarefas realizadas em um único lugar, que não teriam qualquer benefício com o uso de vários agentes;
- Podem se beneficiar do uso de diversos agentes: alguns parâmetros de medida podem ser aplicados, como o desempenho e a confiabilidade, para se notar a vantagem do uso de diversos agentes, na execução de uma tarefa.

Os autores ainda apresentam uma taxonomia para robôs coletivos, baseada em algumas dimensões:

- Tamanho do grupo: o número de agentes no grupo – um ou vários;
- Alcance da comunicação: a distância máxima do alcance da comunicação entre dois elementos do grupo – nenhum, próximo ou qualquer destino;
- Topologia de comunicação: a forma com que a comunicação é realizada entre os componentes do grupo – *broadcast* (difusão para todos), *unicast* (para um endereço específico), em árvore (obedecendo uma hierarquia) ou grafo (com ligações redundantes entre os elementos);
- Largura de banda da comunicação: o limite quantitativo de dados que podem ser transmitidos entre os robôs – livre (sem custo), custo proporcional, custo alto ou sem comunicação;
- Reconfiguração do grupo: a taxa na qual o grupo pode ser reorganizado – estático, reorganização coordenada ou dinâmica;
- Habilidade de processamento de cada unidade: o modelo computacional utilizado por cada elemento do grupo – unidade de soma não linear, autômato finito, autômato a pilha ou Máquina de Turing;

- Composição do grupo: refere-se à homogeneidade/heterogeneidade dos elementos do grupo – idênticos, homogêneos ou heterogêneos.

Em seguida, são apresentados exemplos de classificação de alguns casos conhecidos de aplicações cooperativas para robôs móveis. Tomando-se o exemplo do Futebol de Robôs, sua classificação pela taxonomia, nas respectivas dimensões, seria: vários robôs, com comunicação para qualquer destino, em *broadcast*, com custo proporcional, apresentando reorganização dinâmica, processamento baseado em Máquina de Turing, e composição homogênea.

2.3. COOPERAÇÃO E ROBÔS MODULARES

Sistemas robóticos modulares possuem, como principal característica, a vantagem da reconfiguração de suas partes (módulos). A inserção, a remoção ou o reposicionamento dos módulos permitem que o conjunto assumam formas diferentes e adapte-se a situações ou necessidades do ambiente. Com isso, o mesmo conjunto de módulos pode ser utilizado em diferentes aplicações que envolvem, principalmente, a locomoção. Cada módulo é normalmente dotado de capacidades individuais de processamento, sensoramento, comunicação e mobilidade. Dependendo da implementação permite-se, ou não, o uso de módulos diferenciados, ou heterogêneos. Mas, de forma geral, as habilidades e funcionalidades do sistema são alcançadas pelo arranjo de módulos, ou seja, um módulo sozinho pode demonstrar pouca utilidade.

Talvez o PolyBot² (YIM; DUFF; ROUFAS, 2000), desenvolvido pela Xerox PARC, seja um dos mais conhecidos sistemas modulares de robôs. Em sua mais recente versão, cada módulo, no formato de um cubo, possui um pequeno motor, um processador, memória, sensores de posição e de proximidade, além de comunicação através de uma rede CAN (*Controller Area Network*). Duas faces opostas de cada módulo dispõem de contatos para acoplamento físico e condução elétrica para alimentação e sinalização da rede. Um módulo diferente também compõe o sistema, e dispõe de capacidades de suprimento de energia elétrica (onde uma fonte de suprimento é ligada), além de permitir o acoplamento de módulos em suas seis faces.

² Site do projeto: <http://www2.parc.com/spl/projects/modrobots/chain/polybot/>

Yim, Zhang e Duff (2002) descrevem três tipos diferentes de robôs que possuem a capacidade de alterar o próprio formato. O primeiro deles é o tipo cadeia (*chain*), e define que segmentos de módulos são acoplados ou desacoplados de cada vez. O segundo é o tipo treliça (*lattice*), que descreve os sistemas que mudam de formato de acordo com a movimentação de módulos em uma grade (*grid*) virtual. O último deles refere-se a robôs móveis reconfiguráveis, e permite que cada módulo seja desacoplado e mova-se independentemente.

O PolyBot tem remodelagem baseada em acoplamento de segmentos, e exemplifica o primeiro tipo. Os segmentos de módulos assumem formas que se assemelham a uma cobra, a uma aranha ou a uma esteira rolante, adequadas à locomoção em diferentes tipos de terrenos. Os *swarm-bots* apresentados por Mondada et al. (2005) são bons exemplos do terceiro (último) tipo; tais robôs possuem capacidades individuais de locomoção, sobre rodas e esteiras. Ao se acoplarem em segmentos estendem tais capacidades, atuando em conjunto na transposição de obstáculos e terrenos acidentados. O sistema M-TRAN³ (MURATA et al., 2002), apesar de ser descrito pelos autores como um sistema reconfigurável de robôs, é composto de módulos com capacidades individuais restritas de locomoção. Eles seguem um padrão simétrico de acoplamento, como em uma grade 3D, qualificando o segundo tipo. Os módulos são compostos por duas partes semi-cilíndricas, ligadas por um eixo, como em uma rótula; em uma delas residem as funções de processamento e, na outra, a capacidade motora.

É importante considerar que a cooperação entre os módulos é dependente do tipo de arranjo desejado, ou seja, as partes cumprem papéis definidos de acordo com o objetivo geral (coletivo) a ser alcançado. Apesar de serem permitidas diversas variações de formas, a maior parte dos experimentos descreve, como aplicação, a locomoção. Porém, a aplicação de estratégias distribuídas na cooperação de robôs industriais modulares é um assunto pouco abordado (YIM et al., 2007).

Analogamente, um robô manipulador pode ser decomposto e analisado em cada parte individual que o compõe. Tais partes, apesar de distintas, apresentam capacidades individuais de movimentação, sensoramento e até de comunicação. Portanto, um manipulador pode ser classificado como um sistema *n*-modular, onde *n* é o número de tipos diferentes de módulos que, neste caso, é maior que dois.

³ Site do projeto: <http://unit.aist.go.jp/is/frrg/dsysd/mtran3/>

2.4. CONTROLE DISTRIBUÍDO APLICADO À ROBÓTICA

Tradicionalmente, o modelo de controle aplicado a robôs móveis é decomposto em uma seqüência de módulos (BROOKS, 1986), conforme apresentado na Figura 1.

Figura 1 – Modelo tradicional de controle aplicado a robôs móveis⁴

Os dados sobre modificações do ambiente são captados pelos sensores e interpretados pelo módulo de percepção. A interpretação dos dados permite a geração do modelo do ambiente, inerente ao domínio da aplicação. O modo de planejamento pode então decidir sobre que tarefas deverão ser executadas, em resposta à situação modelada. A partir daí, as tarefas são escalonadas por um módulo de execução, e traduzidas em ações de controle para os motores do robô.

Pode-se notar que o modelo tradicional impõe certa precedência operacional entre os módulos. Isso gera grande dependência do correto funcionamento de cada módulo, ou seja, a paralisação ou mal-funcionamento de um deles pode comprometer todo o sistema.

2.4.1. Controle baseado em comportamento

Uma abordagem diferente para o controle de robôs móveis, baseada em comportamento, é então introduzida por Brooks (1986). O modelo apresentado é composto por diversas camadas (níveis) de abstração, a exemplo de estratégias de decomposição semelhantes, aplicadas em outras áreas de estudo, como Redes de Computadores (TANENBAUM, 2003) e Engenharia de Software (SOMMERVILLE, 2003).

Os oito níveis da arquitetura são organizados do mais baixo (0) ao mais alto (7), e apresentados na Figura 2:

- Nível 0 – Desviar de objetos: evitar e desviar-se de colisões;
- Nível 1 – Perambular: movimentar-se aleatoriamente;

⁴ Adaptada de Brooks (1986)

- Nível 2 – Explorar: explorar o mundo, observando o ambiente;
- Nível 3 – Construir mapas: mapear o ambiente e planejar rotas;
- Nível 4 – Monitorar mudanças: notar e comunicar mudanças no ambiente;
- Nível 5 – Identificar objetos: reconhecer tipos distintos de objetos e associá-los a tarefas específicas e predefinidas;
- Nível 6 – Planejar mudanças: formular planos de ações para modificar o estado do ambiente;
- Nível 7 – Identificar comportamento de objetos: adequar os planos formulados de acordo com o comportamento de outros objetos (atores) no ambiente.

Figura 2 – Modelo de controle baseado em comportamento⁵

As diferentes camadas da arquitetura estabelecem interações através de interfaces predefinidas. De forma geral, e a exemplo de arquiteturas de redes de computadores, cada camada presta serviços para a camada imediatamente superior. Dessa forma, a implementação do sistema deve obedecer ao caminho de “baixo para cima”. Cada camada adicionada acima provê um nível adicional de controle, que pode interferir no sinal resultante, emitido aos atuadores.

A forma adotada por Brooks (1986) para implementar o sistema de controle, consistiu em associar a cada camada uma máquina de estado finito e algumas variáveis de instância. Além disso, a comunicação entre os módulos de *software* é realizada por meio de troca de mensagens. As mensagens são previamente definidas e podem descrever argumentos de entrada e de retorno. Um exemplo de interação entre os níveis é também apresentado pelo autor (BROOKS, 1986).

⁵ Adaptada de Brooks (1986)

Tal abordagem assemelha-se ao paradigma da orientação a objetos, cuja utilização prática não era muito comum na época da publicação desse artigo. Segundo o paradigma, um objeto detém um determinado estado, indicado pelos valores assumidos por seus atributos internos (variáveis de instância). O estado interno de um objeto pode influenciar diretamente o seu comportamento em relação ao ambiente que está inserido. A comunicação entre objetos é realizada pela troca de mensagens, que podem conter parâmetros de entrada ou de retorno. A aplicação do paradigma de programação orientado a objetos no projeto de sistemas de controle é amplamente discutida por Jobling et al. (1994).

A comunicação entre as diferentes camadas de *software* da Figura 2 é assíncrona, ou seja, cada entidade de *software* (ou módulo) pode executar sua função de forma independente das demais. Se alguma entidade tornar-se indisponível, apenas as funções a que ela responde não seriam executadas. Tal abordagem, permite a implementação do controle em partes fracamente acopladas, como componentes de *software* em sistemas distribuídos.

2.4.2. Controle distribuído baseado em agentes

Nwana (1996) classifica a arquitetura de Brooks (1986) como um agente reativo, que não possui um modelo prévio interno sobre o ambiente onde atua, mas que age em resposta a estímulos que dele obtém. De certa forma, não existe um modelo global bem definido no relacionamento entre esses agentes, o que pode levá-los a agir de forma não determinística. No entanto, pode-se notar que o comportamento reativo está associado à forma com que Brooks implementou seus robôs, e não ao modelo que propôs.

A arquitetura em camadas InteRRap, apresentada por Fischer, Müller e Pischel (1996), propõe a criação de agentes híbridos. Dois agentes distintos, um para a base de conhecimento e um para a unidade de controle, permitem que a arquitetura apresentada na Figura 3 implemente um comportamento deliberativo (além do reativo). A forma deliberativa indica a criação e manutenção de um modelo global do ambiente, sobre o qual são definidas regras que influenciarão o comportamento do sistema. Nota-se também que cada agente assume papéis distintos em diferentes camadas ou contextos: baseado em comportamento, planejamento local e planejamento cooperativo.

Figura 3 – Arquitetura híbrida deliberativa-reativa⁶

2.4.3. Controle local *versus* controle global

Em ambas as arquiteturas apresentadas (nas Figuras 2 e 3) pode-se notar que as camadas mais baixas referem-se à solução de problemas locais, e que, conforme se caminha em direção às camadas mais altas, gradativamente o enfoque direciona-se para um contexto global. A distinção entre controle global e local, assim como as diferentes medidas (e requisitos) de desempenho e de confiabilidade para cada um dos dois contextos, são abordadas e tratadas na arquitetura ALLIANCE (PARKER, 1998) para controle de multi-robôs móveis.

Wang, Nakano e Takahashi (2003) descrevem o sistema BeRoSH, que une características de sistemas centralizados e descentralizados em uma solução para controle distribuído de robôs móveis. O conhecimento local é mantido distribuído e descentralizado, para cada agente, mas parte dessas informações integra uma base de conhecimento comum centralizada. Tal base de dados é utilizada para o planejamento e o controle globais. A distinção entre os dois contextos, global e local, é também aplicada ao planejamento da locomoção em sistemas modulares, como o M-TRAN (MURATA et al., 2002). Um planejador global decide sobre o movimento global do conjunto, ou fluxo, descrito em um espaço 3D. Já o planejador local, decompõe os comandos globais de movimentação em passos, baseando-se em um repositório de regras, que incluem esquemas de movimentação.

⁶ Fonte: Fischer, Müller e Pischel (1996)

As múltiplas camadas de um modelo de controle podem ser analisadas quanto a duas dimensões distintas que afetam o serviço que prestam: capacidade computacional e tempo de resposta. A capacidade computacional refere-se ao conjunto de recursos necessários ao processamento e à transformação de dados. O tempo de resposta que um sistema pode oferecer qualifica-o (ou não) para uma aplicação, perante especificações de níveis desejados de determinismo.

Pode-se notar que as camadas mais baixas dos modelos de controle distribuído dedicam-se a tarefas locais. Tais tarefas envolvem a interpretação de leituras realizadas em sensores locais, o planejamento de ações perante regras muito simples, e um tempo de resposta muito pequeno (da ordem de milissegundos), característico de aplicações de tempo real. Nesse nível de atuação, portanto, a demanda por capacidade computacional é baixa e as restrições de tempo de resposta são normalmente críticas.

Já nas camadas mais altas, aspectos globais relacionados ao comportamento do sistema são analisados e decisões estratégicas tomadas. As decisões devem envolver a análise de um grande volume de informações de estado de todo o conjunto, obtidas do processamento de dados de sensoramento do ambiente, e armazenadas em repositórios. As regras aplicadas nesse tipo de decisões também são mais complexas e numerosas, aspectos esses que indicam a necessidade de maior capacidade computacional, tanto em processamento como no armazenamento de dados. Por outro lado, as restrições de tempo para que essas decisões surtam efeito são bem mais flexíveis, e podem permitir tempos de resposta próximos ao de sistemas online (da ordem de segundos).

A mesma relação pode ser notada em níveis intermediários, se for admitido que a demanda por capacidade computacional cresça gradativamente quanto mais alta a camada analisada. Já a demanda por um menor tempo de resposta aumenta em sentido contrário.

O uso de comunicação baseada em mensagens assíncronas é um aspecto muito importante de integração da arquitetura de controle. A estratégia permite que diferentes *softwares* comunicantes de camadas distintas mantenham sua cadência específica de execução, e assim atendam os requisitos individuais de desempenho. Outra vantagem do sistema é permitir a execução simultânea dos diversos módulos fracamente acoplados.

Dada a distribuição de papéis nesse tipo de arquitetura e o isolamento funcional pela estratégia de comunicação apresentada, podem-se definir os perfis de alguns elementos que a compõem. As camadas mais baixas, dados os requisitos de desempenho apresentados, adequam-se aos sistemas embarcados atualmente disponíveis no mercado. As camadas mais altas, por exigirem maior capacidade computacional e permitirem tempos maiores de resposta, podem ser implementadas em computadores remotos, bastando que existam meios de comunicação adequados à integração do sistema. Com a futura evolução do *hardware* embarcado, mais funções de alto nível podem ser para ele migradas.

2.4.4. Controle distribuído aplicado a robôs manipuladores

Como apresentado anteriormente, os robôs manipuladores cumprem importantes funções em aplicações industriais. Apesar de existirem diversas propostas de arquiteturas para modelos distribuídos de controle em robótica, pode-se notar que geralmente se aplicam a soluções de problemas na área de robótica móvel. Portanto, muito pouco se abordou na literatura pesquisada sobre a aplicação de estratégias distribuídas para a solução de problemas de controle em robôs manipuladores.

Lau e Ng (2005) descrevem um *framework* de controle distribuído baseado no paradigma de multi-agentes, aplicado a robôs manipuladores modulares hiper-redundantes. Tal solução inspira-se no modelo imunológico de seres vivos para descrever uma arquitetura de controle baseada em comportamentos.

A definição de uma arquitetura adequada ao controle distribuído de robôs manipuladores consiste parte da solução do problema abordado neste trabalho. Dessa forma, um novo modelo para o controle cooperativo entre partes de um robô manipulador será proposto em momento oportuno, quando da proposta da solução do problema.

2.5. CONSIDERAÇÕES FINAIS

Neste capítulo discorreu-se sobre o assunto da cooperação entre robôs. Uma revisão dessa área de estudos, bem como as principais linhas de pesquisa nela contextualizadas foram apresentadas. A cooperação foi abordada em ambientes de multi-robôs e também em robôs modulares.

Pode-se notar que o estado atual da pesquisa científica nessa área de estudos não oferece sustentação completa ao problema ora estudado. Isso porque os modelos tradicionalmente utilizados para a concepção do ambiente são compartilhados apenas pelos robôs (ou módulos) envolvidos nas tarefas cooperativas. Sendo assim, faz-se necessária a definição de um modelo que possa ser compartilhado entre robôs e seres humanos, assunto abordado no próximo capítulo.

Também foram apresentadas neste capítulo diferentes estratégias de controle de robôs, do modelo tradicional ao adaptativo (reativo e/ou deliberativo), assim como arquiteturas de controle distribuído para robôs móveis. Nota-se que o modelo adaptativo em camadas, proposto por Brooks (1986), é mais adequado à concepção de arquiteturas distribuídas, ideais para o controle de múltiplos agentes. Porém, tal modelo é puramente reativo, o que não atende à demanda pelo comportamento deliberativo desejada para a solução do problema proposto.

Portanto, pode-se por ora concluir que os modelos de controle disponíveis na literatura não atendem plenamente a integração das regras cooperativas e colaborativas, tema principal deste trabalho. Sendo assim, um novo modelo de controle distribuído, que permita a deliberação baseada em tais regras, é proposto e apresentado neste trabalho, no capítulo 5.

3 ATIVIDADES MENTAIS HUMANAS E A COMPUTAÇÃO

De acordo com o apontado no capítulo anterior, a colaboração depende de um modelo adequado à representação do ambiente e dos objetos nele contidos. Tal modelo deve permitir o compartilhamento entre atores robóticos e humanos que integrarão o ambiente colaborativo. Sendo assim, a concepção de um modelo abstrato do funcionamento da mente humana torna-se muito importante para a sua adequada representação pelo sistema computacional, principalmente quando se tem a intenção de reproduzir comportamentos deliberativos semelhantes em um robô.

Para tal propósito, este capítulo apresenta uma visão geral sobre o estudo da mente humana e as suas diferentes estratégias de abstração em ambientes computacionais. Apesar de não figurar entre os objetivos experimentais deste trabalho, a percepção dos seres humanos é descrita na primeira seção. A próxima seção descreve a interpretação e suas abordagens distintas de representação interna: conexionista e simbólica. Na mesma seção é apresentada a linguagem, seus formalismos e abstrações. Nas seções seguintes descrevem-se a consciência, os fundamentos para a relação mente-cérebro e a Teoria Computacional da Mente.

3.1. INTRODUÇÃO

A mente humana é, sem dúvidas, um dos sistemas naturais mais complexos. A ela são atribuídas particularidades de cada ser humano e até a noção de sua própria existência no mundo. A mente é, portanto, consequência e criação do próprio corpo ou, mais especificamente, do cérebro humano.

Muitos filósofos tentaram apresentar explicações convincentes sobre a relação mente-cérebro (FODOR, 1981), sendo consenso que a mente representa o imaterial, enquanto que o cérebro é constituído de matéria. Uma das teorias atribui à mente um aspecto funcional do cérebro, e pela qual se acredita que a predisposição topológica das conexões neuronais cria caminhos ou circuitos capazes de desempenhar funções específicas. Pinker (1998) afirma que tais padrões de conexões são transmitidos hereditariamente, o que consiste parte importante da evolução e adaptação da espécie humana.

Essa mesma abordagem define que os módulos funcionais da mente operam representações internas, abstraídas do mundo real. Neste ponto, duas abordagens ganharam a atenção da ciência: o connexionismo e o simbolismo. Enquanto a abordagem connexionista estabelece uma forma de representação interna inspirada em equivalentes físicos dos neurônios, a simbólica defende a utilização de símbolos de uma linguagem interna (PINKER, 2002) para o mesmo fim.

Ambas as abordagens apresentam vantagens e limitações. A connexionista, se por um lado não representa fielmente a organização e conexões sinápticas dos neurônios verdadeiros e está muito distante das dimensões reais de um cérebro, por outro tem fornecido soluções computacionais eficientes a problemas de percepção, identificação e reconhecimento (de objetos, p.e). A simbólica, por sua vez, não oferece soluções simples para problemas pouco ou mal definidos, porém é mais adequada aos processos mentais que dependem da representação e operação de símbolos de uma linguagem, normalmente relacionados às atividades mentais de níveis superiores. Segundo Minsky (1990), o ideal é que os sistemas computacionais tirem proveito das vantagens de ambas as abordagens.

3.2. A PERCEPÇÃO

A percepção consiste de capacidades que os organismos vivos desenvolveram e adaptaram ao longo dos tempos e que permitem a obtenção de informações sobre o estado do ambiente em que vivem. Segundo Ittelson (1973), a percepção pode ser vista como uma única e imediata resposta de um sistema nervoso a um estímulo do ambiente. De forma geral, os organismos são munidos de sistemas sensoriais, como a visão, o olfato, a audição, o tato e o paladar, que permitem perceber variações de energia que podem indicar mudança de estado das coisas no mundo ao redor.

Durante o processo evolutivo, os seres vivos definiram comportamentos distintos para as diversas situações percebidas do ambiente, o que consistiu a base para a adaptação e posterior transmissão a seus descendentes de tais características hereditárias. Sendo assim, pode-se dizer que o comportamento de um indivíduo depende do repertório de respostas que ele gerou para um conjunto particular de estímulos obtidos do ambiente. Para Ittelson (1973), até a existência cultural, assim como a individual, dependem da percepção.

Apesar de ser geralmente tratada como uma experiência consciente de algum sentido, o processo da percepção é muito complexo. Organismos vivem em ambientes amplos, dos quais recebem informações de contexto, pelos órgãos sensoriais, e as processam juntamente com informações de outras fontes, em um sistema nervoso. A recepção e o processamento da informação do ambiente constituem a área de estudo da percepção (ITTELSON, 1973).

Ainda segundo Ittelson (1973), os organismos, de acordo com sua complexidade, apresentam diferentes demandas ao sistema sensorial. Indivíduos mais simples, ou primitivos, podem utilizar a percepção do ambiente para evitar situações de perigo ou para se alimentarem, enquanto os mais complexos podem associá-la a outras funções, como: localização, identificação, comunicação, atração sexual e outras relações sociais mais complexas. Seres humanos também utilizam tais capacidades em apoio a outros processos, como, linguagem e estética. Dessa forma, a percepção e a cognição fazem parte de um sistema maior, cuja função é processar informação.

O modelo geral da percepção dos organismos vivos tem sido reproduzido (simulado) por máquinas. Analogamente, diversos tipos de dispositivos sensores, tais como óticos, ultrassônicos e térmicos, permitem captar variações de energia do ambiente e representá-las em sinais úteis para processamento por um sistema eletrônico. Às variações de níveis desse sinal no tempo são associados padrões que permitem, de certa forma, diferenciar situações específicas e previstas para as mudanças de estado do ambiente onde o sistema está inserido.

3.2.1. O sistema visual

Dos diversos sentidos utilizados pela percepção, a visão é o mais importante. Ela permite a recepção de sinais luminosos provenientes de objetos remotos, enquanto que para o tato faz-se necessário o contato físico com eles. Além disso, o sistema visual não necessita absorver parte do objeto analisado como ocorre no processo palativo. Outra vantagem da visão é a de não ser intrusiva, ou seja, o indivíduo que observa não necessita emitir energia ao ambiente na maioria das situações (quando já existe luz emissiva ou difusa no ambiente), o que não ocorre com sonares presentes em diversos animais.

De forma geral, a luz estimula os fotorreceptores, que estimulam os nervos óticos, e estes produzem a experiência visual (ITTELSON, 1973). De acordo com Bear, Connors e Paradiso (2002), a percepção visual é a tarefa de identificação e associação de significado a objetos no espaço. Os objetos, vistos no mundo real, têm formato (geometria), cor (e textura), posição (e orientação) e podem mover-se.

Segundo Bear, Connors e Paradiso (2002), a retina não só passa adiante a informação sobre padrões de luz que obtém, mas também extrai informações sobre a imagem focalizada. O conjunto de informações extraídas e transmitidas pela retina denomina-se projeção retinofugal. A projeção retinofugal se propaga a partir da retina para (e nessa ordem) o nervo ótico, o quiasma ótico e o trato ótico.

Como ilustrado na Figura 4, a visão obtida do olho direito forma o campo visual (hemisfério) direito, mas também se estende para parte do hemisfério esquerdo; de forma semelhante ocorre com o olho esquerdo. Essa região central de intersecção presente em ambas as projeções denomina-se retina nasal.

Figura 4 – Modelo geral da projeção visual⁷

⁷ Fonte: <http://www.edoctoronline.com/medical-atlas.asp>

No quiasma, os nervos óticos se cruzam: os sinais provenientes do olho direito, referentes à porção visual do hemisfério direito, são encaminhados ao trato ótico esquerdo, juntamente com os sinais da retina nasal obtidos pelo olho esquerdo; os sinais obtidos pelo olho esquerdo, referentes ao campo visual esquerdo, são encaminhados ao trato ótico direito, juntamente com os sinais da retina nasal obtidos pelo olho direito. A maior parte das conexões neuronais que deixam o trato ótico segue para o núcleo geniculado lateral (NGL), do tálamo dorsal. No NGL são conectados os neurônios que farão a projeção para o córtex visual primário, denominada radiação ótica.

As informações obtidas da retina pelas células ganglionares são mapeadas em uma estrutura bidimensional, localizada na superfície do colículo superior. Tal organização denomina-se projeção retinotectal, e constitui uma importante característica da visão, denominada retinotopia. Existem muito mais células ganglionares com campos receptores dentro (ou perto) da fóvea do que na periferia. No colículo superior, um arranjo de neurônios ativados por um ponto de luz comanda os movimentos dos olhos e da cabeça para projetar a imagem desse ponto no espaço na fóvea, através de conexões indiretas com neurônios motores no tronco cerebral. Dessa forma, o estímulo luminoso é detectado e tratado pelo sistema ótico.

Os olhos desviam-se de um ponto a outro do campo visual, delimitado pela posição da cabeça. As imagens nítidas se produzem na retícula da fóvea e, fora dela, a visão perimetral é muito limitada. Sendo assim, não se podem obter detalhes precisos de forma, cor e iluminação de objetos localizados fora do ponto focal (PINKER, 1998).

Ainda em Bear, Connors e Paradiso (2002), afirma-se que o processamento de informações visuais, que descreve a atividade de reconhecimento de objetos e suas partes em uma imagem, é integrado em três canais interdependentes. O primeiro denomina-se canal M (ou canal magnocelular), é composto por células simples e, por estar conectado a neurônios sensíveis à seleção de orientação e direção, além de campos receptivos binoculares, é especializado na análise motora de objetos. O segundo, canal P-IB, é composto de células complexas binoculares, insensíveis ao comprimento de onda da luz, mas muito seletivas a estímulos de orientação, permite discriminar e identificar objetos, baseado em seus moldes (formatos). O terceiro, chamado de canal *blob*, é especializado na análise da cor dos objetos.

Em Coren, Ward e Enns (1999) define-se campo visual como o total de luz do ambiente que estimula os olhos. Essa luz refletida forma a imagem na retina, que consiste de uma distribuição bidimensional de várias intensidades e comprimentos de onda. Existem, porém, quatro aspectos gerais do ambiente que afetam a visão:

- Fonte de luz: refere-se à direção e à intensidade das regiões que produzem (emitem) luz no ambiente;
- Propriedades refletivas: descrevem as propriedades das várias superfícies que sofrem o contato com a luz;
- Orientação da superfície: descreve a inclinação da reflexão da luz em referência a uma linha imaginária perpendicular à superfície (normal);
- Posição do observador: especifica a relação entre o olho do observador e a cena.

Ainda em Coren, Ward e Enns (1999), descrevem-se as diferentes formas (*shapes*) como regiões com brilho uniforme dentro do campo visual. As formas são separadas das demais formas e do fundo de cena pelos seus contornos. Um contorno pode ser descrito como uma região na imagem formada na retina onde a intensidade (ou cor) da luz muda abruptamente. Já a textura visual descreve-se por coleções de minúsculos elementos de contorno ou formas que não se diferem da média do brilho ou cor daquela região.

Além disso, as formas podem apresentar outros atributos, ou características (*features*), como cor, textura e profundidade, além de movimento. O processo de obtenção de tais atributos denomina-se extração de características. Nosso sistema visual, então, opera na produção de objetos perceptuais do arranjo de *blobs* e de contornos contidos na distribuição de luz na retina. Sendo assim, os objetos perceptuais são referências mentais criadas a partir de vários níveis de processamento e interpretação aplicados à imagem da retina.

Para Pinker (1998), a visão permite produzir, a partir de imagens do mundo externo, informações que são úteis para quem vê. Inicialmente, o sistema visual localiza o alvo, ou seja, deve definir onde termina o objeto e começa o fundo da cena. Trata-se de um problema inerente à definição das bordas e limites dos objetos visualizados. Para resolver o problema da visão em profundidade, o cérebro então compensa e corrige a visão distorcida e cria uma interpretação próxima do que julga mais conveniente. Além disso, como os olhos são projetados para um ponto focal comum os campos visuais são sobrepostos, mantendo-se a paralaxe binocular.

Segundo Coren, Ward e Enns (1999), outro problema relacionado à identificação de objetos é o agrupamento das formas, ou seja, como as formas são combinadas mentalmente para serem futuramente reconhecidas ou identificadas. Para tanto, nosso cérebro vale-se de algumas estratégias, descritas pelas leis da Proximidade, da Similaridade, da Contigüidade, do Movimento Comum e do Fechamento.

O cérebro ainda procura quantificar os padrões mais comportados de figuras, ou seja, existe uma predisposição em se obter objetos que apresentam simetria vertical. As figuras ditas “boas” para análise pelo cérebro apresentam um padrão simétrico e regular. O reconhecimento de objetos significa a experiência de perceber algo previamente conhecido. Por outro lado, a identificação de objetos envolve a denominação de um objeto, corretamente classificado de acordo com um esquema predefinido, sabendo-se previamente em que contexto é usualmente encontrado e também as suas relações com outros conceitos.

O processo de reconhecimento de objetos, em linhas gerais, pode ser realizado em duas estratégias distintas. A primeira, orientada a dados, envolve o uso de um conjunto fixo de regras e procedimentos a serem aplicados a todos os dados de entrada. A outra, orientada a conceitos, envolve a aplicação de processos conceituais de alto nível, como memória de experiências passadas, estratégias de organização, e expectativas baseadas no conhecimento do mundo e eventos prévios ou no contexto envolvido. O reconhecimento conceitual orientado a expectativas contextuais permite compensar a falta de detalhes na extração de características orientada a dados.

O processo de identificação de objetos envolve três fases distintas. Durante o primeiro estágio de percepção da informação a imagem da retina é processada para se detectarem os contornos e as características simples, que serão agrupadas em figuras e regiões em um segundo estágio. No terceiro estágio, as partes extraídas das imagens são comparadas com memórias de objetos perceptuais para se determinar a identidade dos estímulos. Expectativas e contexto têm um papel muito importante no estágio de comparação da entrada com as memórias; a correspondência necessária à identificação não precisa ser boa, pois se pode esperar por um objeto específico, devido ao contexto ou a intenção da busca realizada, mas não o encontrar.

Para Pinker (1998), a partir da detecção, o cérebro tenta criar uma identificação do objeto, valendo-se de três tipos distintos de análise: pela forma, pela cor e pela luminosidade. Em seguida, assume a representação mais provável, de acordo com expectativas prévias (contexto) e as evidências registradas. Três teorias são aplicadas, de forma geral, ao processo de identificação de objetos (COREN; WARD; ENNS, 1999):

- Pandemônio: para cada estágio da análise dos padrões da imagem de entrada define grupos especialistas de *demons* (ou módulos). Os primeiros, chamados *image demons* (módulos de imagem) passam o conteúdo da imagem da retina para conjuntos de *feature demons* (módulos de características). Estes localizam as características e encaminham o resultado para os *cognitive demons* (módulos cognitivos), cada qual espera por uma combinação específica de resultados enviados pelos *feature demons*. Um *decision demon* pode ser envolvido no processo toda vez que se chegue a um impasse;
- Baseada em modelos: envolve a comparação da projeção do objeto real na retina com uma projeção do modelo (*template*) que foi armazenado previamente na memória;
- Baseada em componentes: os objetos percebidos são analisados quanto às partes distintas que os compõem. As partes são comparadas com um conjunto padronizado de componentes, ou *geons*. Os *geons* definem formas padrões e suas variações, como cubos, esferas, cones e outras. A idéia geral consiste em descrever objetos em termos dos seus componentes *geons* e então comparar com outras descrições baseadas em *geons* previamente armazenadas em memória.

De acordo com Pinker (1998), apesar de representar os modelos de objetos tridimensionalmente na memória, a visão humana é bidimensional. O processo de reconhecimento permite descrever os objetos representados na superfície visível. Em linhas gerais, o cérebro analisa posições e ângulos das partes do objeto visualizado, alinhando um referencial sobre seus eixos de extensão e de simetria. A seguir, procura associar suas partes (ou todo o objeto) às representações em memória. Um fato interessante é que o córtex visual é acessado quando pensamos em imagens criadas na memória, em vez de obtê-las pelos órgãos visuais.

3.2.2. Visão computacional

Segundo Shapiro e Stockman (2001), o objetivo da visão computacional é permitir a tomada de decisões úteis sobre objetos reais e cenas, baseadas em imagens obtidas (sentidas). Além disso, aplica-se à construção de descrições de cenas, a partir de imagens. Sendo assim, pode-se dizer que a visão computacional (ou artificial) serve importantes métodos e ferramentas para conceber interfaces a sistemas com o ambiente em que atua. Tais interfaces são implementadas com dispositivos óticos de entrada (como câmeras e sensores óticos) que, assim como os olhos, obtêm sinais luminosos do meio externo.

Os mesmos autores ainda apresentam os principais elementos de sistemas computacionais aplicados à visão artificial:

- Sensoriamento: define como os sensores obtêm as imagens do mundo;
- Informação codificada: uma vez obtida, a imagem deve ser entendida quanto à sua geometria, textura, mobilidade e tipo de objetos envolvidos;
- Representação: define modelos para a descrição dos objetos e suas partes, propriedades e relações;
- Algoritmos: métodos para processamento das informações de imagens e para construção de descrições do mundo (virtual) e seus objetos.

Dentre as diversas aplicações da visão computacional, encontram-se representações de imagens do corpo (medicina), de mapas e dados de satélites, e construção de cenas para orientação de robôs móveis. Para tanto, vários problemas ainda são enfrentados, principalmente aqueles relacionados à detecção de bordas de elementos em imagens, recuperação de detalhes, combinação e computação de características de imagens distintas, e extração de representações não iconográficas. Este último ramo citado define métodos e modelos para aferição e comparação de características mensuráveis de imagens, ou de suas partes.

3.3. A INTERPRETAÇÃO

A interpretação pode ser entendida como um processo mental subsequente à percepção humana. Com ela, pode-se atribuir significado às representações internas associadas aos objetos e sinais percebidos no ambiente externo. Dessa forma, portanto, a mente humana cria sua própria representação do mundo externo, onde se insere.

Esclarecer como o cérebro humano é capaz de gerar tal representação interna tem sido o foco do estudo das neurociências e da psicologia. Segundo Teixeira (1997), a psicologia, durante muito tempo, esteve em busca de um paradigma que lhe fornecesse fundamentação metodológica suficiente à sua validação como uma ciência. A partir de um esforço interdisciplinar de diversas áreas do conhecimento, como computação, psicologia, lingüística, filosofia e neurologia, fundamentou-se a criação da ciência cognitiva, cuja missão principal consiste em simular computacionalmente os processos mentais humanos.

O processo e o modelo utilizados para a geração da representação interna (mental), porém, variam de acordo com a abordagem utilizada. Duas abordagens distintas foram propostas e estudadas ao longo do tempo para a concepção do modelo computacional da mente: simbolista e conexionista (TEIXEIRA, 1997).

Na década de 1940, um movimento denominado cibernética inspirou pesquisadores a buscarem uma representação da atividade psicológica na forma de modelos matemáticos. Vislumbrava-se a possibilidade de se criarem circuitos elétricos que pudessem modelar o funcionamento do cérebro. A inspiração para tal vinha da analogia entre sistema nervoso e circuitos elétricos. Sendo assim, para se simular as atividades cerebrais, um computador deveria implementar um *hardware* específico, que permitisse computações paralelas (como os neurônios naturais). Infelizmente, a arquitetura proposta por Von Neumann, presente na grande maioria dos computadores modernos, fornece iguais tratamentos a dados e programas e opera seqüencialmente. Tais idéias influenciaram diretamente a concepção do conexionismo, um movimento que despertou grande interesse da pesquisa na década de 1960. A idéia principal reside na adoção de redes neurais artificiais como modelos computacionais capazes de simular as atividades neuronais. *Perceptrons*, em redes de neurônios com duas camadas, foram então concebidos para a solução de problemas que envolviam o reconhecimento de padrões visuais (percepção).

Nas décadas seguintes, ganhou a cena um novo movimento que se propôs a modelagem da mente como um conjunto de representações de tipo simbólico e regidas por um conjunto de regras sintáticas. Esse movimento deu origem à concepção da Inteligência Artificial Simbólica, e encontrou forças nas críticas às limitações dos *perceptrons* desferidas no final da década de 1960 por Marvin Minsky, a exemplo da operação “ou exclusivo” (XOR) que estes não podiam realizar.

Anos mais tarde, em Minsky (1990), é apresentado o então atual estado da arte das abordagens simbolista e conexionista. Ambas as visões apresentavam suas vantagens em áreas que, muitas vezes, não se sobrepõem. Enquanto o conexionismo apresentava bons resultados em aplicações de imitação de comportamento animal, reconhecimento de padrões, otimização e localização de dados, podia-se dizer o mesmo do simbolismo em sistemas que envolvem decisão baseada em metas, análise causal e *parsing*. Por outro lado, ambas as abordagens apresentavam desvantagens e pouco progresso para serem exclusivamente consideradas como estratégias únicas para a representação das atividades mentais humanas. Sendo assim, o autor vislumbrou que sistemas que tirem proveito das duas abordagens poderiam apresentar resultados melhores no futuro.

3.3.1. O conexionismo

Como mencionado anteriormente, o termo conexionismo concebe um movimento pelo qual se buscou representar computacionalmente as atividades mentais com uso de redes neurais artificiais. Tal estratégia descreve uma abordagem de construção *bottom-up* (MINSKY, 1990), que consiste em definir inicialmente pequenas unidades de programas, que operam princípios lógicos e matemáticos básicos para simular cada célula do córtex cerebral. A seguir, ao se realizar a interconexão e a operação conjunta dessas unidades básicas, podem ser obtidos os comportamentos e operações mais complexos.

Segundo Haykin (1998), um neurônio é uma unidade básica de processamento de informações para uma rede neural. A Figura 5 mostra um diagrama que representa um modelo básico de neurônio. Nele, podem ser notados os principais elementos que constituem um neurônio: um conjunto de sinapses (com pesos associados), uma função acumuladora e uma função de ativação.

Figura 5 – Modelo básico de um neurônio

Neurônios podem ser interconectados, de forma que o sinal de saída de um seja aplicado à entrada de outro(s). Diversos formatos de redes neurais podem ser obtidos, mas de maneira geral podem ser classificados em uma das três categorias: de camada única e sem retroalimentação; de múltiplas camadas e sem retroalimentação; recorrentes. O tipo de arquitetura de uma rede neural implica diretamente no tipo de algoritmo de treinamento a ser utilizado.

Perceptrons podem ser considerados como aplicações mais simples de redes neurais para a classificação de padrões linearmente separáveis. Consistem basicamente de um único neurônio com pesos sinápticos e *bias* ajustáveis. O algoritmo de classificação utilizado pelos *perceptrons*, de acordo com o teorema de convergência apresentado por Rosenblatt, posiciona uma superfície de decisão, na forma de um hiperplano, entre as ocorrências de padrões (contidos em vetores) em duas classes distintas. Dessa forma, cada *perceptron* constituído de um único neurônio é capaz de, depois de treinado, decidir sobre problemas de classificação binária, isto é, que envolvem duas classes distintas (ou hipóteses).

3.3.2. O simbolismo

O simbolismo caracteriza-se por um movimento que defende a representação simbólica das atividades mentais, de forma parecida à realizada por computadores. Para tanto, essa abordagem define uma estratégia construtiva do tipo *top-down* (MINSKY, 1990): inicia-se com concepções de senso comum no nível da psicologia e a partir daí tenta-se imaginar os processos capazes de simulá-las.

A idéia básica de tal estratégia consiste em partir de um desafio maior e buscar a solução pela sua decomposição em partes mais simples. Isso denota a técnica reducionista, uma abordagem típica utilizada em IA (Inteligência Artificial), chamada de programação heurística.

De acordo com Teixeira (1997), o paradigma da IA Simbólica introduz que a simulação da inteligência não depende de máquinas com *hardware* específico, mas pode ser obtida de programas de computador que operariam sobre dados ou representações. Segundo o mesmo paradigma, o funcionamento da mente assemelha-se ao de uma Máquina de Turing, pois opera informações representadas na forma de símbolos, e tais símbolos podem ser combinados pelo uso de um conjunto de regras de formação.

A abordagem simbólica justifica-se pela argumentação de que o ser humano é o único, dentre os outros seres até hoje conhecidos, capaz de desenvolver as chamadas atividades cognitivas superiores, como a Matemática e a Linguagem (TEIXEIRA, 1997). Ambas as atividades são extremamente dependentes de representação simbólica. Portanto, a inteligência exclusivamente humana, capaz de desenvolver as atividades cognitivas superiores, resultaria do encadeamento de representações mentais, ou símbolos. Sendo assim, a mente poderia ser comparada a um programa de computador e, para replicá-la basta que se encontre um programa de computador que permita simulá-la.

As relações entre os símbolos que representam conceitos de objetos e demais elementos percebidos no ambiente são convenientemente representadas pelo uso de proposições (PINKER, 1998, p. 316):

Quando sai de cena a visão e entra o pensamento, não há como contornar a necessidade de símbolos abstratos e proposições que selecionem aspectos de um objeto para a mente manipular.

3.3.2.1. Linguagem e significado

A linguagem é um importante veículo pelo qual as pessoas podem externar os seus pensamentos. De um modo geral, linguagem e processos mentais estão intimamente ligados e pode-se dizer que a mente cria a linguagem (PINKER, 2002, p. 62). Porém, quando alguém recorda de algo que leu geralmente lembra-se das principais idéias e não das palavras exatas. Dessa forma, pode-se dizer que existe

um modelo interno mental mais adequado à representação de coisas e fatos. Esse modelo, referido por Pinker (2002) como “mentalês” (linguagem da mente), é capaz de abstrair aspectos do mundo real, como espaço, tempo, objetos, quantidade, velocidade, causalidade e categorias.

Chomsky (2009) descreve a representação interna da linguagem por meio de estruturas profundas. Tais estruturas são livres de ambigüidades e especificidades de línguas faladas e podem ser associadas a significados. As estruturas que se relacionam à vocalização e à sintaxe das linguagens naturais são conhecidas como superficiais. Sendo assim, uma estrutura profunda única pode produzir diferentes representações equivalentes em estruturas superficiais.

As línguas naturais, como o Português, não são adequadas à representação sistemática de idéias ou significados. Existem cinco problemas principais, apontados por Pinker (2002, p. 89-92), que impedem a associação direta de termos em línguas naturais a conceitos. O primeiro é a ambigüidade, notada na atribuição de diferentes significados ou interpretações para uma mesma palavra. O segundo problema é a falta de explicação lógica, que impede a determinação sistematizada do bom senso.

O terceiro problema é o da co-referência, pelo qual, diferentes termos ou palavras podem ser utilizados para se referir a algo ou alguém. O quarto relaciona-se ao contexto, pois muitos aspectos de uma linguagem têm interpretação dependente de uma situação (o que os lingüistas descrevem como “dêixis”). O quinto e último problema é a sinonímia, ou seja, diferentes arranjos de palavras podem representar o mesmo significado.

Por outro lado, a descrição formal de uma gramática que permita gerar uma linguagem não ambígua, pode ser utilizada como forma de representação interna, tanto para uma pessoa, como para um sistema computacional. Tal idéia é defendida por Pinker (2002, p. 83) onde se argumenta que a Máquina de Turing, um formalismo hipotético criado por Turing, realiza transformações simbólicas sobre uma linguagem interna, assim como a mente humana. E ainda, seguindo-se o mesmo raciocínio, tanto um cérebro como um computador pode ser composto por um grande número de representações e processadores sofisticados; dotando-se o cérebro de olhos para visualizar e músculos para agir sobre o ambiente, tem-se um indivíduo funcional; da mesma forma, com câmeras e mecanismos a um computador, obtém-se um robô (PINKER, 2002, p. 89).

3.3.2.2. Linguagens formais

Formalmente, uma linguagem pode ser definida como uma coleção de cadeias de símbolos de comprimentos finitos. Estas cadeias são denominadas sentenças da linguagem e são formadas pela justaposição de elementos individuais, os símbolos ou átomos da linguagem. A formalização de uma linguagem depende de uma gramática, que especifica os elementos e regras que compõem suas sentenças.

Segundo Chomsky (2009), uma gramática pode ser descrita como um sistema de regras que especifica a relação som-significado em uma determinada língua. Uma gramática, dita gerativa de uma língua, é capaz de gerar um conjunto infinito de descrições estruturais, que consistem abstrações de objetos de sons e significados particulares. Sendo assim, as gramáticas podem ser consideradas dispositivos formais de geração de sentenças das linguagens que definem, e podem ser caracterizadas por quádruplas ordenadas, conforme equação 1.

$$G = (V_n, V_t, P, S) \quad (1)$$

Onde:

- V_n vocabulário não terminal da gramática G : conjunto de todos os elementos simbólicos utilizados para as leis de formação;
- V_t vocabulário terminal: conjunto de símbolos ou átomos dos quais as sentenças da linguagem são constituídas;
- P conjunto de todas as leis de formação que definem a linguagem: cada construção parcial, representada por um não-terminal, é definida como um conjunto de regras de formação (produção) relativas à definição do não-terminal a ela referente. Cada produção P tem a forma:

$$\alpha \Rightarrow \beta \quad \forall \alpha \in (V_n \cup V_t)^+; \beta \in (V_n \cup V_t)^*$$
 α é uma cadeia que contém ao menos um não-terminal;
- S símbolo inicial (axioma) da gramática: não-terminal (elemento de V_n) que dá início ao processo de geração de sentenças.

Os diferentes tipos de linguagens podem ser classificados de acordo com as gramáticas que as produzem. Em linhas gerais, uma gramática estabelece restrições formais para a criação dos termos para uma determinada linguagem, a partir de um alfabeto previamente estabelecido. Noam Chomsky propôs tal organização em quatro diferentes classes, conhecida na literatura como Hierarquia de Chomsky (Figura 6).

Figura 6 – Hierarquia de Chomsky

Os níveis da Hierarquia de Chomsky são descritos por $n \in \{0, 1, 2, 3\}$, que indica as regras de restrição da gramática. Quanto maior o valor de n , mais restrita é a gramática (gramáticas regulares são do nível três, p.e). Uma gramática de qualquer tipo maior é também pertencente a um nível menor; o inverso nem sempre é verdadeiro. Analogamente, uma linguagem do tipo n é caracterizada pela existência de alguma gramática do tipo n que a descreva, podendo ser também classificada como sendo do tipo menor, se for o caso.

As Gramáticas com Estrutura de Frase (tipo zero) são irrestritas, ou seja, nenhuma restrição é imposta e produzem Linguagens Irrestritas. Suas produções obedecem a forma descrita pela equação 2.

$$a \Rightarrow b \quad \forall a \in (V_n \cup V_t)^+ ; b \in (V_n \cup V_t)^* \quad (2)$$

As Gramáticas Sensíveis ao Contexto (nível um) definem apenas uma restrição, a de que nenhuma substituição reduza o comprimento da forma sentencial à qual a substituição é aplicada. Tais gramáticas geram Linguagens Sensíveis ao Contexto e suas produções obedecem às formas descritas pelas equações 3 ou 4.

$$a \Rightarrow b \quad \forall a, b \in (V_n \cup V_t)^+ \mid |a| \leq |b| \quad (3)$$

$$bAg \Rightarrow bag \quad \forall A \in V_n; a \in (V_n \cup V_t)^+; b, g \in (V_n \cup V_t)^* \quad (4)$$

As Gramáticas Livres de Contexto (tipo dois) condicionam-se às substituições das regras definidas pelas produções. Elas geram Linguagens Livres de Contexto e obedecem a forma geral, descrita pela equação 5.

$$A \Rightarrow a \quad \forall A \in V_n; a \in (V_n \cup V_t)^* \quad (5)$$

As gramáticas do tipo dois podem ser representadas pela Forma Backus-Naur (BNF – *Backus-Naur Form*) (BACKUS et al., 1963), pois o lado esquerdo de cada regra é composto por um único símbolo não-terminal. As diferenças restringem-se ao nível sintático e o procedimento geral envolve a substituição do símbolo $\Rightarrow$ por $::=$ e a colocação do prefixo $<$ para cada não-terminal. Além disso, as várias opções de uma regra de produção são separadas pelo símbolo $|$.

As Gramáticas Regulares (nível três) são de grande importância para o estudo de compiladores por possuírem propriedades adequadas para a obtenção de reconhecedores simples. Elas geram Linguagens Regulares (ou Expressões Regulares), cujas produções são restritas às formas descritas pela equação 6.

$$A \Rightarrow aB; A \Rightarrow b \quad \forall A, B \in V_n; a \in V_t; b \in V_t \cup \{\lambda\} \quad (6)$$

3.3.2.3. A Máquina de Turing

Segundo Diverio e Menezes (2000), Alan Turing propôs um modelo abstrato de computação, conhecido como Máquina de Turing. O modelo, definido por um método formal, foi aplicado por Turing para explorar os limites de uma máquina para expressar soluções de problemas. Outros modelos, publicados na mesma época, como as máquinas de Norma e de Post apresentaram resultados semelhantes, reforçando as propostas de Turing.

Os resultados semelhantes apresentados por trabalhos independentes, em termos de suas capacidades de expressar computabilidade, reforçam a Hipótese de Church ou Hipótese de Turing-Church: "A capacidade de computação representada

pela Máquina de Turing é o limite máximo que pode ser atingido por qualquer dispositivo de computação". Dessa forma, uma Máquina de Turing consiste de um formalismo capaz de simular qualquer computador.

A inspiração de Turing para a sua máquina foi reproduzir uma situação na qual uma pessoa, possuidora de um instrumento de escrita e um apagador, realiza cálculos em séries de quadrados, dispostos seqüencialmente em uma fita de papel. Inicialmente, a folha contém apenas os dados do problema. A partir de então, a pessoa pode: ler um símbolo de um quadrado; alterar um símbolo em um quadrado; direcionar seus olhos para outro quadrado vizinho; terminar os cálculos, quando for obtida uma possível resposta.

Figura 7 – Representação esquemática da Máquina de Turing

A Figura 7 ilustra o esquema geral da Máquina de Turing, idealizada por Turing. Seus elementos são formalmente definidos, mas podem ser entendidos com a idéia desse esboço. A fita representa os dispositivos de entrada, saída e memória de trabalho; ela é finita à esquerda e tão grande quanto o necessário à direita. A fita é dividida em células seqüenciais que permitem armazenar individualmente um único símbolo. Os símbolos são elementos do alfabeto de entrada, do alfabeto auxiliar (interno), β (branco) ou $<$ (marcador de início de fita).

Outro elemento do modelo é a unidade de controle, que reflete o estado corrente da máquina. Para descrevê-lo, a unidade dispõe de um número finito e predefinido de estados. Além disso, possui uma unidade de leitura e de gravação (cabeça da fita), que permite o acesso a uma célula por vez. A cabeça lê o símbolo na célula sob ela e grava ali um novo símbolo. Após esse processo, move-se para a célula vizinha da direita ou da esquerda.

Um programa (ou função) de transição completa o esquema da máquina. Tal programa comanda as leituras e gravações, o sentido de movimento da cabeça e define o estado da máquina. Para tanto, o programa vale-se do estado atual da máquina e do símbolo lido.

Formalmente, a Máquina de Turing é definida como uma 8-upla (equação 7).

$$M = (\Sigma, Q, \sqcap, q_0, F, V, \beta, \lhd) \quad (7)$$

Onde:

- Σ alfabeto de símbolos de entrada;
- Q conjunto finito de estados possíveis da máquina;
- $\sqcap$ programa ou função de transição (é uma função parcial):
 $\sqcap: Q \times (\Sigma \cup V \cup \{\beta, \lhd\}) \rightarrow Q \times (\Sigma \cup V \cup \{\beta, \lhd\}) \times \{E, D\}$
 E e D são, respectivamente, Esquerda e Direita;
- q_0 estado inicial da máquina, tal que q_0 é elemento de Q ;
- F conjunto de estados finais, tal que F está contido em Q ;
- V alfabeto auxiliar;
- β símbolo especial branco;
- $\lhd$ símbolo especial marcador de início da fita.

O símbolo de início de fita ocorre exatamente uma vez e sempre na célula mais à esquerda da fita. A função, ou programa ($\sqcap$), considera a existência do estado corrente $p \in Q$, e um símbolo lido da fita $a_u \in (\Sigma \cup V \cup \{\beta, \lhd\})$. Dessa forma, o programa pode então determinar o novo estado da máquina $q \in Q$, e também o símbolo que será gravado na fita $a_v \in (\Sigma \cup V \cup \{\beta, \lhd\})$, assim como o sentido do movimento da cabeça $m \in \{E, D\}$. O programa pode ser representado por um grafo finito $\sqcap(p, a_u) \rightarrow (q, a_v, m)$, conforme mostra a Figura 8.

Figura 8 – Programa representado por um grafo finito

Uma aplicação para os modelos formais de máquinas é a de reconhecer cadeias de símbolos como pertencentes, ou não, a uma determinada linguagem. A Máquina de Turing é a única especificação formal para uma máquina universal que apresenta recursos suficientes para o reconhecimento de linguagens irrestritas, produzidas por Gramáticas com Estrutura de Frase (tipo zero).

3.3.2.4. O Teste de Turing e a contestação de Searle: A sala chinesa

De acordo com Teixeira (1997), Alan Turing publicou em seu artigo “Computação e Inteligência” um formalismo que, segundo o qual, permitia decidir se uma máquina pode pensar. O formalismo, que passou a ser conhecido como Teste de Turing, baseou-se no que ele chamava de “jogo da imitação”. O teste consistia em fazer com que um interlocutor conversasse com uma pessoa e com um computador, por meio de um terminal de dados (tela e teclado). O interlocutor deve então submeter perguntas, que serão respondidas pela pessoa e pelo computador. Caso o comportamento do computador não possa ser distinguido do apresentado pelo humano, então se afirma que ele pode pensar.

O teste, na realidade, condicionava a habilidade de pensar à simples comparação com o comportamento humano na mesma situação. Como nem os filósofos conseguiam chegar a um consenso sobre o que é pensar, por muito tempo o Teste de Turing forneceu aos cientistas um método adequado para se descobrir a presença de tal faculdade.

Em 1980, porém, John Searle apresentou contestações ao que se poderia considerar inteligência em uma máquina, com seu Argumento do Quarto Chinês. O exercício consiste em um falante, que não fala Chinês, trancado em um quarto. O falante recebe um texto escrito em Chinês, um conjunto de regras de transformação escritas em Português, que permitem realizar operações sobre o texto em Chinês. Tais operações descrevem a decomposição e a recomposição de palavras com base em um roteiro. O procedimento também prevê o reconhecimento de palavras-chave na pergunta recebida, e a busca por elas no texto em poder do falante, que então será decomposto e recomposto de acordo com as regras de transformação.

As contestações do Argumento do Quarto Chinês demonstram uma vulnerabilidade do Teste de Turing. O sistema (composto pelo quarto, falante, texto e regras) poderia passar pelo Teste de Turing, sem que o falante pudesse compreender as sentenças escritas em Chinês. Dessa forma, o Teste de Turing pode se consistir de uma condição necessária, porém não suficiente, para atribuição de estados e faculdades mentais a um organismo ou máquina.

3.4. A CONSCIÊNCIA

Segundo Minsky (2006), o termo “consciência” admite significados múltiplos. É muito comum encontrar evidências de seu uso para diferentes tipos de processos mentais e para diferentes propósitos. Dentre tais associações figuram sentimentos, emoções e pensamentos. A maioria das atividades mentais geralmente ocorre inconscientemente, ou seja, não se necessita de sua ciência para que ocorra (como o comando motor de músculos e a recepção dos sentidos). Por outro lado, os processos mentais ditos de “alto nível” apresentam certas características peculiares:

- Fazem uso da memória mais recente: as decisões e deliberações dependem da associação de lembranças recentes, como algo que acaba de acontecer, ou alguma coisa que foi imaginada;
- Operam muito mais em modo serial do que paralelo: a maioria dos sistemas vitais, durante a evolução humana, foi associada a módulos mentais específicos que podem ser operados paralelamente. Por outro lado, as atividades superiores (ou de alto nível), que envolvem o raciocínio e a deliberação são muito mais recentes e ainda demandam da operação no módulo mental principal. Dessa forma, toda a atenção mental é direcionada a um processo superior por vez;
- Utilizam descrição abstrata, simbólica e verbal: os processos mentais de níveis mais baixos, como a percepção e o reconhecimento de padrões, são operados a grandes velocidades e podem ser facilmente representados por redes de neurônios (abordagem conexionista). Por outro lado, os processos de níveis superiores demandam a reflexão. Ou seja, fatos e objetos, devidamente representados por um modelo interno abstrato, são confrontados e comparados durante processos conscientes. A organização dos modelos internos (símbolos) é realizada em redes semânticas, que guardam entre eles relações de significado, e não meramente pesos e valores;
- Usam modelos para representar a si mesmo (o ser pensante): as experiências vivenciadas por seres humanos muitas vezes contextualizam o próprio ser (o que ele sentiu ou como reagiu naquela situação). Tais experiências são representadas pela mente com modelos abstratos de sua própria existência.

Mas, em Koch e Tononi (2008) são apresentadas importantes considerações sobre a relação da consciência com atividades humanas e seus estados mentais, comprovadas por exames clínicos. De acordo com o estudo, estar consciente não depende da interação do indivíduo com o ambiente. Além disso, a consciência também não requer outras capacidades, como emoção, atenção, memória (explícita ou de trabalho), auto-reflexão e linguagem (fala).

Dessa forma, pode-se concluir que a consciência retém um significado que extrapola a simples associação às atividades mentais superiores. Em outras palavras, não há consenso nas diversas áreas de estudo que expressem significado completo e amplamente aceito para esse termo. Sendo assim, sua representação por modelos computacionais ainda é ambígua e pouco aplicável.

3.5. A RELAÇÃO MENTE-CÉREBRO

Durante muito tempo, diferentes concepções sobre as atividades cognitivas humanas têm dividido os cientistas de diversas áreas do saber, como psicologia e neurociência. De forma geral, os psicólogos tendem a explicá-las pela representação da mente como algo abstrato (imaterial), já os neurocientistas o fazem pelo estudo das conformações físicas (materiais) do próprio cérebro.

As filosofias tradicionais da mente podem ser divididas em duas categorias distintas, as teorias dualistas e as teorias materialistas. Mas, nas últimas décadas, e com o advento das pesquisas em Inteligência Artificial, uma nova abordagem – nem dualista ou materialista – para a filosofia da mente emergiu, o funcionalismo (FODOR, 1981).

3.5.1. O dualismo

A abordagem dualista defende que a mente é composta de uma substância abstrata (não física), diferentemente do cérebro que é material (físico). Todas as coisas do mundo podem ser classificadas em uma das categorias: mental ou física. O principal problema com o dualismo é sua impossibilidade de conceber coerentemente a interação entre o mental e o físico, ou seja, não torna claro como uma mente abstrata pode gerar um efeito físico sem violar as principais leis da Física, como conservação de massa, energia e momento (FODOR, 1981).

Em Teixeira (1997), são apresentadas variações para a vertente dualista:

- Cartesianismo: postula que mente e corpo são substâncias distintas, sendo a primeira inextensa e imaterial, e a segunda extensa e material;
- Interacionismo: defende que a mente afeta o corpo vice-versa;
- Paralelismo: sustenta que o corpo e a mente correm em paralelo, nada tendo um a ver com o outro;
- Epifenomenismo: sustenta que o corpo afeta a mente, mas a mente não afeta o corpo. Sendo assim, a mente é um subproduto da atividade cerebral que produz experiência subjetiva, mas é causalmente inerte.

3.5.2. O materialismo

Ainda segundo Fodor (1981), nas teorias materialistas, o mental não se distingue do físico, em vez disso, defendem que todos os estados mentais, suas propriedades, os processos e as operações envolvidos são idênticos aos equivalentes físicos. Duas linhas derivam do materialismo: o comportamentalismo (*behaviorismo*) e as teorias de identidade do estado central.

Os comportamentalistas afirmam que as relações de causa-efeito podem ser classificadas como comportamentos de resposta a estímulos do ambiente. As teorias comportamentalistas podem ser classificadas em dois grupos: comportamentalismo radical e comportamentalismo lógico.

- O comportamentalismo radical rejeita a existência de estados mentais, propriedades, processos e operações, ou seja, que todo o comportamento não depende de causalidade mental, mas sim, da simples resposta a estímulos;
- O comportamentalismo lógico é uma visão semântica sobre o significado dos termos mentais, que podem ser expressos em sentenças de controle, no formato “if-then”. De certa forma, procura suprir a causalidade mental com regras de construção em tal formato, que podem ser disparadas caso os estímulos correspondentes sejam detectados no ambiente.

Os principais problemas com esta abordagem advêm de sua incompatibilidade de aplicação às atividades cognitivas superiores e da própria refutação dos processos mentais de causa-efeito. Encadeamentos de eventos obrigatoriamente implicam em relações de causa-efeito entre estados mentais, fato ignorado pelo comportamentalismo lógico. Ou seja, de uma forma geral, as duas abordagens comportamentalistas (radical ou lógica) pregam a inexistência dos estados mentais (FODOR, 1981).

A teoria da identidade do estado central estabelece uma equiparação entre os eventos mentais, estados e processos com eventos neurofisiológicos (TEIXEIRA, 1997). A propriedade de um determinado estado mental é idêntica à propriedade de se estar em um dado estado neurofisiológico.

3.5.3. O funcionalismo

Mas, para que o materialismo pudesse ser amplamente aceito pelos filósofos da mente, um dilema deveria ser resolvido: se de um lado, a teoria da identidade está certa sobre o caráter causal das interações entre mente e corpo, por outro lado, o comportamentalismo lógico está correto no caráter relacional das propriedades mentais.

A resposta para esse dilema foi a concepção de uma nova abordagem, o funcionalismo. O funcionalismo não pode ser classificado como uma teoria dualista, nem uma teoria materialista, pois surgiu de reflexões filosóficas no desenvolvimento da inteligência artificial, teoria computacional, lingüística, cibernética e psicologia (coletivamente chamadas de ciências cognitivas).

O funcionalismo introduz o conceito do papel causal, pelo qual um estado mental pode ser descrito pelas suas relações causais com outros estados mentais. A base do funcionalismo é a distinção que a ciência da computação estabelece entre *hardware* (componentes físicos) e *software* (programas). Sob esta visão da psicologia, um sistema pode descrever um ser humano ou uma máquina e a sua constituição básica (neurônios ou componentes eletrônicos) não é o que realmente importa, mas como suas partes estão organizadas (FODOR, 1981). Dessa forma, o funcionalismo não descarta a possibilidade de um sistema mecânico ou eletrônico possuir estados mentais e processos.

A noção de que diferentes sistemas físicos podem apresentar a mesma funcionalidade é muito semelhante à definição de máquinas formais na Teoria da Computação, a exemplo do formalismo da Máquina de Turing apresentado neste capítulo. Máquinas definem conjuntos de estados que podem ser assumidos, de acordo com os símbolos que recebem como entrada e o estado que anteriormente ela descrevia. Na realidade, as máquinas consistem bons exemplos para os dois conceitos centrais do funcionalismo: que os seus estados mentais são inter-relacionados (como em relações causais) e que as máquinas podem ser realizadas em muitos sistemas. Além disso, de acordo com esta abordagem, a descrição da mente como um *software* a isenta da necessidade de neurônios.

3.6. TEORIA COMPUTACIONAL DA MENTE

Segundo Pinker (1998), além de apoiar as faculdades como ver, pensar, sentir e interagir, a mente dedica-se a atividades superiores, como a arte, a religião e a filosofia. Tais atividades superiores não podem ser apoiadas por teorias que descrevem as atividades mentais humanas como reações diretas a eventos notados no ambiente. Sendo assim, grande parte da atividade cognitiva é dependente de iterações mentais, ou seja, de encadeamentos de relações causa-efeito que dependem de e contribuem com as alterações entre os diferentes estados internos mentais de um ser humano. A mente, segundo uma concepção apresentada em Pinker (1998, p. 32), pode ser descrita como:

... um sistema de órgãos de computação, projetados pela seleção natural para resolver os tipos de problemas que nossos ancestrais enfrentavam em sua vida de coletores de alimentos, em especial entender e superar em estratégia os objetos, animais, plantas e outras pessoas.

Ao interpretar a sentença, o autor analisa individualmente as afirmações que a compõem. A mente é o que o cérebro faz (ou pelo menos é parte das atividades que realiza), pois processa informações através do pensamento, e pensar caracteriza-se como computação. A mente organiza-se em módulos (órgãos) mentais que são especialmente concebidos para cada área de interação com o mundo.

A lógica básica dos módulos é descrita por nosso programa genético e, sendo assim, constituem-se de natureza inata e hereditária. O funcionamento dos módulos foi moldado pela seleção natural a fim de resolver os problemas

enfrentados por ancestrais humanos, e as características funcionais dos indivíduos mais adaptados foram transmitidas aos seus descendentes.

O autor também argumenta a hereditariedade das estruturas inatas da mente fundamentada em observações relatadas sobre gêmeos idênticos. Os gêmeos idênticos compartilham características comuns, como hábitos e vontades, mesmo quando são criados em ambientes distintos, fisicamente isolados. Portanto, segundo a Teoria Computacional da Mente, o cérebro (material) cria a mente (imaterial) por meio de modelos funcionais de conectividade neuronal, que são transmitidos hereditariamente.

Em Goetz e Shacelford (2007) é descrita a área de estudos da psicologia evolucionária. A seleção natural permitiu que lentamente os circuitos do cérebro humano fossem moldados para a solução de problemas adaptativos enfrentados por nossos ancestrais. Para cada problema adaptativo específico foi necessário produzir um comportamento adequado para a sua solução. Dessa forma originaram-se os módulos de processamento de informações, também descritos como mecanismos psicológicos evoluídos.

Os mecanismos psicológicos evoluídos podem ser entendidos como modelos capazes de transformar uma entrada (interna ou externa) em uma saída (comportamento, atividade psicológica ou entrada para outro mecanismo psicológico), de acordo com regras de decisão (procedimentos do tipo *if-then*). A seleção natural não pode operar diretamente o comportamento humano, mas sim os genes associados aos mecanismos psicológicos que o produzem. Os genes são os moldes utilizados para sua auto-replicação, capazes de determinar o fenótipo, até certa parte, pela transcrição gênica.

Deve-se considerar também o efeito da Regulação Epigenética, pelo qual, diversos fatores como os ambientais, os alimentares e os emocionais modificam a transcrição e, portanto, a expressão gênica (GEBARA, 2009). Pode-se, portanto, afirmar que as experiências vivenciadas pelos seres humanos, seus hábitos e suas interações com o meio ambiente potencialmente afetam a transcrição e a replicação dos genes e conseqüentemente o fenótipo (inclusive o psicofenótipo) de uma pessoa em vida; traços dessas adaptações genéticas podem ser transmitidos aos seus descendentes.

3.6.1. Limitações da abordagem conexionista

Outra motivação para a concepção de uma teoria da mente advém das dificuldades para se modelar as atividades cerebrais humanas com a abordagem conexionista. Em Teixeira (1997) são apresentados dois problemas que fundamentam essas dificuldades: o problema da descrição e o problema epistêmico.

O problema da descrição está relacionado às dimensões envolvidas na modelagem. De acordo com Koch e Tononi (2008), o córtex cerebral humano possui aproximadamente 100 bilhões de neurônios e centenas de trilhões de sinapses ou conexões. Deve-se considerar também que os diferentes estados mentais envolvem ativações neuronais em áreas descontínuas do córtex, e que a plasticidade do cérebro humano permite a substituição de neurônios danificados (ou até áreas inteiras do córtex) por outros, em lugares distintos. A análise combinatória das conexões neuronais associada ao aparente indeterminismo da plasticidade cerebral não permite sequer definir a complexidade do problema envolvido, quanto mais, tratá-lo.

Para se ter uma pequena idéia do que pode vir a ser tal complexidade, em Adler (2008) descreve-se um trabalho com o mapeamento 3D das conexões neuronais ativas de uma mosca das frutas (gênero *Drosóphila*), para os seus vários estados mentais. Para tanto, utiliza-se de um microscópio eletrônico de varredura capaz de gerar um bilhão de pixels por segundo e, conseqüentemente, produz próximo de um *petabyte* (1000 *terabytes*) de dados por dia.

O problema epistêmico relaciona-se ao fato de que uma descrição completa do cérebro será sempre mais complexa que ele próprio. Essa concepção pode transcender a capacidade do próprio cérebro humano, pois até hoje não se pôde provar que o cérebro humano seja capaz de tratar um problema de tal dimensão e, sendo assim, ele também não poderia compreender e reconhecer como verdadeiro algo mais complexo que si. Portanto, pode-se por ora afirmar que não é possível replicar todo o cérebro em sistemas artificiais.

Em Lebedev e Nicolelis (2006) é apresentado o estado da arte da pesquisa em interfaces cérebro-máquina (BMI – *brain-machine interfaces*) para as abordagens invasivas e não-invasivas. Técnicas não-invasivas, como o eletroencefalograma, são mais seguras, mas fornecem amostras com qualidade limitada, seja pela fidelidade do sinal, como na sua frequência; já as técnicas invasivas

sujeitam o indivíduo a lesões físicas e infecções, mas permitem obtenção de sinal mais fidedigno. Em seguida, os autores apresentam um estudo de caso aplicado em macacos: sensores foram colocados no córtex cerebral, mais especificamente na região responsável pelo controle da atuação motora, de onde obtiveram a leitura do sinal elétrico emitido por grupos de neurônios; tais sinais foram utilizados para ativar um braço robótico remotamente, permitindo uma realimentação sensorial ao macaco pelas imagens de vídeo exibidas a ele.

Os próprios autores afirmam que a atual tecnologia utilizada pelos sistemas que operam interfaces cérebro-máquina não os permite da utilização de sinais cognitivos que tratam de representações neuronais de alta ordem dos movimentos. McFarland e Wolpaw (2008) também sugerem tal solução, combinada com a resolução de metas por cinemática inversa, para as atuais limitações das BMIs no controle de múltiplos graus de liberdade. Sendo assim, pode-se também afirmar que o estado atual da área impossibilita aplicações que envolvam a leitura de estados mentais, como estratégia para o mapeamento da consciência do indivíduo.

Outro ponto importante a ser evidenciado refere-se à estratégia utilizada para a leitura do sinal. A abordagem baseia-se na obtenção de um sinal composto que representa a média dos sinais individuais emitidos por grupos de neurônios (NICOLELIS, 2001). Sendo assim, o próprio sinal obtido pelo sistema não é representativo para a concepção das conexões neuronais e as respectivas ativações individuais envolvidas, tornando-o insuficiente para a geração de mapeamento físico da rede neuronal em análise.

3.6.2. Oportunidades para a abordagem simbólica

As constatações apresentadas na seção anterior reforçam ainda mais a motivação para o desenvolvimento de uma abordagem alternativa para o estudo da mente. Dois aspectos importantes podem ser destacados da concepção da mente apresentada em Pinker (1998), que constituem o alicerce da Teoria Computacional da Mente. O primeiro deles é que as computações mentais são aplicadas sobre informações, e que estas possuem representação interna simbólica. O segundo aspecto relaciona-se à constituição funcional dos módulos mentais que realizam as computações.

Dessa forma, não importa o tipo de indivíduo (físico) onde a computação mental será realizada, mas o funcionamento dos módulos que compõem essa mente e os símbolos que serão submetidos a ela. Portanto, a informação e a computação residem em padrões de dados e em relações lógicas que são independentes do meio físico que as conduz. Como consequência, crenças e desejos também podem ser vistos como informações, encarnadas como configurações e símbolos.

Ao vislumbrarem a possível representação de estados mentais em máquinas, diversos pesquisadores trazem à cena atual a discussão sobre um tema controverso, a singularidade. Para Vinge (2008), diversos cenários da pesquisa atual apontam para uma nova era da evolução humana, comparável a marcos históricos, como: a descoberta do fogo, o desenvolvimento da agricultura e à Revolução Industrial. Essa nova era seria marcada pelo surgimento abrupto (VINGE, 2008) ou gradual (BROOKS, 2008) de uma entidade (humana ou máquina) com níveis muito superiores de inteligência aos dos atuais seres humanos. Segundo Bimber (2008), alguns futuristas predizem que, ainda no meio deste século, será possível transferir a representação do cérebro humano e seus estados mentais para um supercomputador.

3.7. CONSIDERAÇÕES FINAIS

Como apresentado em diversas oportunidades neste capítulo, ambas as abordagens computacionais, simbólica e conexionista, detêm importantes papéis na representação de modelos mentais. Pode-se notar que as atividades vitais caracterizam-se pela demanda de reação rápida aos estímulos obtidos por sentidos. Tais atividades foram adaptadas ao longo da evolução dos seres vivos e muitas vezes são qualificadas como instintivas, ou seja, não demandam deliberação ou atividades mentais superiores. Também se pode afirmar que são tratadas por módulos mentais altamente especializados e que operam simultaneamente.

Os órgãos que servem de interface para o indivíduo com o ambiente, seja pela obtenção de estímulos por sentidos a eles associados ou pela expressão de ações para alteração do ambiente, se deparam com situações novas continuamente. Ou seja, o ambiente não exhibe condições perfeitas e predefinidas para formatos, cores e disposições de objetos. Além disso, o ambiente pode apresentar ao indivíduo novos objetos e situações jamais experimentadas.

Essa incerteza pode ser tratada por modelos de percepção e reação ajustáveis, uma boa aplicação para a abordagem conexionista. Tal estratégia consiste solução adequada à interação de robôs com o meio ambiente, mediante a identificação e reação a estímulos dele percebidos.

Porém, outras atividades mentais notadamente superiores, que demandam auto-reflexão e deliberação por processos que envolvem relações causa-efeito, são dependentes de representações abstratas internas. Essas representações abstratas permitem a associação de objetos e fatos a elementos formais (símbolos) que compõem uma linguagem interna adequada ao meio onde são armazenadas. O relacionamento entre objetos e fatos armazenados é qualificado por termos e sentenças que expressam significado e não por valores e pesos. Isso torna o modelo conexionista pouco aplicável, e a abordagem simbólica mais adequada.

Como também descrito, as linguagens podem ser expressas por gramáticas formais e interpretadas por máquinas universais. Porém, as linguagens naturais apresentam ambigüidades e são extremamente complexas para a interpretação por máquinas. Dessa forma, e com o propósito de se definir uma representação comum para a colaboração entre robôs e seres humanos, um modelo restrito de formato para expressão de regras faz-se necessário.

Portanto, pode-se concluir que a abordagem simbólica é a mais adequada para o mapeamento das regras em um formato compartilhado entre o agente humano e o computador (que comanda o agente robótico). Em linhas gerais, as regras definem políticas que regem o ambiente colaborativo, e descrevem elementos nele presentes. Tal assunto é abordado em detalhes no próximo capítulo.

4 AMBIENTES COLABORATIVOS

A colaboração neste trabalho é tratada como um caso especial de relação entre entidades em um ambiente, onde pelo menos uma delas é um ser humano. Dessa forma, pode-se então caracterizar ambientes colaborativos por conjuntos de elementos e fatores que se relacionam com a presença de um ser humano. Além disso, em uma relação de colaboração devem-se estabelecer metas comuns aos agentes (atores), que compartilham um mesmo modelo mental do ambiente.

Uma vez que a abordagem adotada neste trabalho consiste em conceber modelos sistematizados, torna-se agora necessário compreender os diferentes elementos de um ambiente colaborativo e então definir uma forma adequada para representá-los, bem como as relações entre eles. Com tal propósito, este capítulo inicia-se com a descrição dos diferentes elementos presentes em ambientes colaborativos, bem como a interação e o formato da comunicação entre eles. Em seguida, são apresentados os recursos que a área da Computação dispõe para permitir a abstração de tais ambientes. Maior atenção é dedicada à descrição do modelo e da linguagem adotados para a formalização dos ambientes colaborativos.

4.1. INTRODUÇÃO

O ser humano abstrai características de coisas ou eventos pela percepção do ambiente onde se insere. Tais características consistem elementos de dados que, após os processos de interpretação e classificação, são correlacionados com conceitos previamente aprendidos e geram novas informações para a base de seu conhecimento. De forma semelhante, os sistemas de informação computadorizados permitem a representação dos elementos de dados e de suas relações para a geração de informação.

Dessa forma, pode-se dizer que o processo geral de abstração de um ambiente real envolve, inicialmente, a abstração de dados, que representam características dos elementos nele inseridos. Em seguida, os dados podem ser correlacionados de acordo com algum critério, para assim comporem informações. Informações combinadas permitem a formação de bases de conhecimento, e estas, por sua vez, a concepção de teorias para sistemas inteligentes.

Para Pinker (1998), um sistema realmente inteligente deve ser equipado com uma pequena lista de verdades essenciais e um conjunto de regras que podem ser utilizadas para se deduzirem as implicações. Mas, para que um sistema também demonstre bom senso, ele deve deduzir apenas implicações relevantes ao seu conhecimento e ao contexto. Sendo assim, para se construir um robô que simule um ser humano, ele deve apresentar, além de órgãos sensoriais (como visão) e coordenação motora, o bom senso e a motivação.

Bom senso e motivação talvez consistam o maior desafio para um robô, pois implicam na simulação da atribuição de princípios e valores, para assim permitirem a adequada representação de suas crenças, desejos e intenções. Por outro lado, os modelos computacionais têm evoluído bastante para apoiar abstrações de sistemas complexos, que consistem os ambientes colaborativos. Os elementos de tais ambientes são discutidos nas próximas seções e, em seguida, são apresentadas áreas de estudo da Computação e algumas técnicas que permitem apoiar a abstração dos mesmos.

4.2. ELEMENTOS DE AMBIENTES COLABORATIVOS

Conforme apresentado no capítulo anterior, Pinker (2002) defende o “mentalês” como o modelo interno que a mente utiliza para abstrair as características do mundo real. Entre os aspectos que podem ser abstraídos, encontram-se as noções de espaço, tempo, objetos, quantidade, velocidade, causalidade e categorias. Tais aspectos consistem a visão do mundo que um ator (ou agente) pode dele abstrair, e que pode ser representada em uma ontologia, descrita adiante neste documento.

Um ator pode ser entendido como um elemento que detém uma abstração do ambiente onde está inserido. Dessa forma, o ator pode perceber o estado de um ambiente, deliberar ações que interfiram e modifiquem esse estado, e também sofrer conseqüências causadas por ações de outros agentes no ambiente. Sendo assim, os atores diferenciam-se de objetos pela capacidade de deliberarem ações que causam mudanças no ambiente, enquanto que os objetos geralmente apenas sofrem as conseqüências de tais ações.

4.2.1. Interação entre elementos em ambientes colaborativos

Os elementos de um ambiente colaborativo também podem ser classificados entre aqueles que afetam esse ambiente, e promovem a alteração de seu estado, e aqueles que não o fazem. Dentre os que promovem alterações no ambiente, existem aqueles que o fazem por decorrência de comportamentos deliberativos ou reativos, e aqui neste trabalho são classificados como atores. Dessa forma, permite-se que tanto um ser vivo, como uma máquina (um robô), participe como ator em um ambiente colaborativo.

A atividade realizada por um ator em um ambiente colaborativo pode afetar o estado de outros atores e objetos participantes do mesmo ambiente. Além disso, tais ações podem ser qualificadas por noções de tempo (instante ou duração) e espaço (dimensão ou local). Tais qualificadores podem, portanto, complementar a relação de causalidade da atividade em um ambiente colaborativo.

A interação pode ser definida como um caso particular de ação, reativa ou deliberativa, que envolve reciprocidade entre atores em um ambiente. Em determinadas circunstâncias, os atores podem tomar ciência da ação realizada por outros atores, deliberar ou reagir aos efeitos dela, e gerar (ou não) uma resposta ao estímulo. Torna-se então necessário o estabelecimento de meios de comunicação, pelos quais os atores receberão e transmitirão estímulos de/para os outros atores; estímulos esses que, em alguns casos, transmitem os sinais que representam os símbolos de uma linguagem.

Neste trabalho, procurou-se entender principalmente a interação entre atores humanos e robóticos em um ambiente colaborativo. Em Drury, Scholtz e Yanco (2003) é apresentada uma taxonomia para classificar a interação entre robôs e seres humanos, com base na concepção que tais agentes detêm dos seus pares. Nessa linha, os autores apresentam cinco componentes da concepção notada nas interações entre robôs e humanos, quando trabalham juntos em uma tarefa síncrona:

- Humano-robô: entendimento que os humanos possuem das localizações, identidades, atividades, estados e arredores dos robôs;
- Humano-humano: entendimento que os humanos possuem das localizações, identidades, atividades, estados e arredores dos demais parceiros humanos;

- Robô-humano: o conhecimento obtido pelos robôs dos comandos dados por seres humanos;
- Robô-robô: o conhecimento obtido pelos robôs dos comandos dados por outros robôs;
- Concepção humana de toda a missão: o entendimento das metas globais definidas para as atividades conjuntas entre robôs e seres humanos, bem como da medida de seus progressos.

Pode-se notar que a taxonomia proposta por Drury, Scholtz e Yanco (2003) reconhece a importância de ambos os tipos de atores, robôs e humanos, manterem informações sobre o ambiente onde atuam. Porém, de certa forma a abordagem dos autores restringe parte do comportamento deliberativo aos atores humanos, pois são os únicos a deterem o entendimento completo da missão dos atores no ambiente. Uma abordagem diferente poderia permitir que tanto os seres humanos como os robôs tivessem noção de toda a missão a ser cumprida, e então compartilhassem as informações sobre os estados do ambiente, pelo uso de um modelo comum para tal representação.

4.2.2. Linguagens e regras

Como descrito no capítulo anterior, linguagens são formalismos que definem conjuntos de símbolos escritos ou falados, e que podem ser utilizados como meios de comunicação. Símbolos escritos podem ser combinados em palavras que, dependendo do contexto, associam-se a significados de idéias que se desejam transmitir. A composição de sentenças de uma linguagem é previamente definida por uma gramática geradora. Dessa forma, sentenças são constituídas por uma seqüência finita de símbolos de algum alfabeto finito (SLONNEGER; KURTZ, 1995).

A sintaxe de linguagens de programação descreve como os símbolos podem ser combinados para se criarem sentenças bem formadas (ou programas). A interpretação das palavras de uma frase fornece o seu significado, ou semântica, uma representação interna adequada ao sujeito onde ocorre. Dessa forma, as intenções sobre um comportamento desejado podem ser descritas pelo uso de regras, compostas por símbolos de uma linguagem. Para se atenderem os propósitos deste trabalho, as regras devem ser expressas em um formato interpretável por robôs e seres humanos.

Gramáticas formais restritas e não ambíguas foram propostas para a criação de linguagens de programação para computadores. A forma Backus-Naur (BNF – *Backus-Naur Form*) é modelo formal amplamente adotada para a especificação de gramáticas que descrevem símbolos terminais de um alfabeto válido, símbolos não-terminais e regras de produção. O maior benefício de uso de linguagens no estilo BNF é a facilidade em se definirem programas interpretadores (léxicos).

4.3. COMPUTAÇÃO APLICADA A AMBIENTES COLABORATIVOS

Na seção anterior buscou-se apresentar os diversos elementos presentes em um ambiente colaborativo. Em linhas gerais, tais ambientes possuem atores, objetos e lugares. Os atores podem realizar atividades que afetam o estado de outros atores e objetos, e tais atividades podem ser qualificadas por relações de tempo e espaço.

Neste trabalho procurou-se adotar uma maneira sistematizada para a representação desses ambientes colaborativos. Sendo assim, foram pesquisados modelos e recursos na área da Computação para abstração de tais elementos. Para tanto, nesta seção são apresentados os conceitos sobre CSCW (*Computer Supported Cooperative Work*), agentes de *software*, ontologias e políticas para ambientes colaborativos.

4.3.1. Trabalho Cooperativo Apoiado por Computador

De acordo com Grudin (2010), o termo CSCW (*Computer Supported Cooperative Work*) significa Trabalho Cooperativo Apoiado por Computador e é usado para designar a pesquisa na área do trabalho em grupo e como os computadores podem apoiá-lo. Essa área de estudos teve sua origem nos primeiros esforços da área de automação de escritório para permitir o acesso simultâneo a processadores de textos e planilhas eletrônicas por um grupo de usuários. Os estudos na área têm apresentado importantes contribuições para a solução de problemas atuais, advindos da integração de conteúdos e pessoas em modelos complexos como o das redes sociais. Uma subárea, denominada CSCWD (*CSCW in Design*), publica pesquisas sobre o projeto de artefatos complexos e sistemas que requerem a cooperação de equipes multidisciplinares.

A integração de atores robóticos e humanos em um ambiente colaborativo consiste um problema complexo. A aplicação de técnicas e modelos de CSCW pode permitir o compartilhamento de uma concepção comum sobre uma tarefa realizada por robôs e humanos em um mesmo ambiente (YANKO; DRURY, 2004). Em Gutwin e Greenberg (2001) descreve-se tal concepção comum como um modelo mental compartilhado por cada membro que interage com o espaço de trabalho cooperativo.

4.3.2. Agentes de *software*

Como mencionado anteriormente, um ator pode ser qualificado como um elemento que detém uma abstração do ambiente onde está inserido. Da mesma forma, agentes de *software* devem armazenar um modelo do ambiente onde operam (HUHNS, 2003). Sendo assim, um ator pode ser abstraído em um sistema computadorizado, através de um agente de *software*.

Em Nwana (1996) encontra-se a definição clássica de agente como um componente de *software* e/ou de *hardware* que é capaz de agir precisamente na realização de tarefas em nome de seu usuário. Seu papel pode ser formalizado por um ator, um tipo especial de objeto que possui algum estado interno encapsulado e pode responder mensagens de outros semelhantes. O autor apresenta cinco diferentes dimensões para a classificação de agentes:

- Mobilidade (móvel ou estático): habilidade de se mover através de uma rede de computadores;
- Tipo de resposta (deliberativo ou reativo): um agente deliberativo possui um modelo simbólico interno, que lhe permite planejar e negociar tarefas com outros agentes a fim de coordenar uma execução. Agentes reativos apresentam comportamento do tipo estímulo/resposta ao estado atual do ambiente no qual está imerso;
- Função: dentre eles a autonomia, a aprendizagem e a cooperação;
- Papel: papel assumido pelo agente na aplicação;
- Hibridismo: agentes que combinam duas ou mais das outras dimensões citadas.

Agentes de *software* podem apresentar características que os qualifiquem em uma ou mais dimensões. Em seu trabalho, Nwana (1996) simplifica a combinação dessas várias dimensões a uma lista de sete tipos de agentes, suficientes para descrever a maioria dos agentes de *software* estudados na época:

- Agentes colaboradores: enfatizam autonomia e cooperação com outros agentes para executar tarefas para seus donos. Exemplos desses agentes incluem agentes baseados no paradigma crenças-desejos-intenções (do inglês, BDI – *belief-desires-intentions*);
- Agentes de interface: enfatizam autonomia e aprendizagem e operam em favor da simplificação do acesso ao sistema por usuários humanos;
- Agentes móveis: consistem programas de computador capazes de “vagar” por redes como a Internet, e interagir com outros sistemas para coleta de informações em prol de seu usuário;
- Agentes de informação: permitem gerenciar, manipular e ordenar informações oriundas de várias fontes distribuídas, a exemplo dos agentes de busca da Web;
- Agentes reativos: não mantêm representação simbólica interna do ambiente e emitem respostas de acordo com estímulos dele recebido;
- Agentes híbridos: tiram proveito de mais de uma dimensão. Por exemplo, um agente que implementa tanto o comportamento reativo, quanto o deliberativo, poderia apresentar respostas mais rápidas aos estímulos do ambiente e também tratar a solução de problemas orientados a metas;
- Agentes inteligentes: agentes capazes de combinar e utilizar todas as características das dimensões apresentadas.

Pelo apresentado nas seções anteriores, pode-se dizer que os ambientes colaborativos são abstraídos por sistemas complexos. Tais sistemas são assim classificados porque devem permitir a representação de atores com comportamentos reativos e/ou deliberativos, além de objetos e qualificadores de tempo e espaço, que compõem o estado geral do ambiente. Portanto, tal cenário consiste de justificativa suficiente para a definição de agentes inteligentes, que necessitam de um modelo comum para a especificação dos conceitos abstraídos.

4.3.3. Ontologias

A palavra ontologia pode assumir definições distintas na literatura científica. Uma delas descreve ontologia como a especificação explícita da conceitualização de objetos (GRUBER, 1993). Em linhas gerais, uma ontologia permite capturar o conhecimento consensual de forma genérica e pode ser reutilizada e compartilhada por componentes de *software* e por grupos de pessoas (GÓMEZ-PÉREZ; FERNÁNDEZ-LÓPEZ; CORCHO, 2004).

Sistemas baseados em conhecimento podem tirar proveito do uso de ontologias para a representação abstrata dos objetos (ou coisas) e das relações existentes entre eles, e definir um vocabulário de termos na língua humana. Portanto, os agentes inteligentes podem compartilhar os conceitos comuns, representados em uma ontologia, para a representação de um ambiente colaborativo.

Huhns e Singh (1997) argumentam que uma ontologia consiste um modelo computacional adequado à abstração de alguma porção do mundo. Os mesmos autores ainda afirmam que uma ontologia expressa um acordo sobre uma conceitualização compartilhada, e implementa a estrutura para a modelagem do conhecimento de um domínio e dos acordos sobre a representação de teorias particulares sobre domínios específicos.

Uma ontologia pode ser fisicamente estruturada como uma rede semântica ou um grafo, onde os nós são conceitos ou objetos individuais e os arcos representam relações ou associações entre os conceitos (STAMPER, 2001). Outros elementos, como propriedades, atributos, limitações, funções e regras, são acrescentados a tal rede para a caracterização e descrição do comportamento dos conceitos. Além disso, ontologias geralmente permitem a representação de vários tipos de relações entre as suas classes, como generalização/herança, agregação e instanciação.

Mais especificamente, na representação gráfica de uma ontologia, cada conceito pode ser tratado como uma entidade ou objeto, enquanto que as normas podem ser utilizadas como especificações de restrições e procedimentos operacionais, programáveis em computadores (STAMPER, 2001). As normas comportamentais podem ser descritas formalmente como regras, com aplicação de operadores deônticos, como “obrigado”, “permitido e “proibido”.

Uma linguagem para especificação de ontologia amplamente difundida é a OWL⁸ (*Web Ontology Language*). A criação de uma ontologia para a Web foi motivada pelo anúncio da Web Semântica no evento “XML2000”, realizado por Tim Berners-Lee, criador da *World Wide Web* e diretor do *WWW Consortium*⁹. Em Berners-Lee, Hendler e Lassila (2001), afirma-se que o poder real da Web Semântica será atingido quando programas que coletam conteúdo Web de fontes diversas processarem a informação e trocarem resultados com outros programas, o que salienta o importante papel dos agentes inteligentes.

A OWL foi concebida como uma extensão das linguagens RDF (*Resource Description Framework*) e XML (*eXtensible Markup Language*). A formalização e a estruturação de documentos da linguagem XML adicionadas à definição de modelos relacionais da linguagem RDF, torna possível a definição de semânticas simples para objetos. A OWL é subdividida em três especificações de acordo com o nível de expressividade semântica:

- OWL *Full*: apresenta maior expressividade, com todos os elementos da linguagem OWL disponíveis, que podem ser usados sem restrição;
- OWL-DL: menor expressividade que *Full*, pois impõe restrições para assegurar que um procedimento de raciocínio deve existir, o que a torna um modelo de linguagem computável;
- OWL *Lite*: versão mais limitada, que não permite algumas operações com conjuntos; apresenta as mesmas restrições que a OWL-DL.

4.3.4. Políticas de colaboração

Políticas de colaboração são normas para as interações em um ambiente que podem ser estabelecidas pelo agrupamento lógico de regras com metas e objetivos bem definidos (EDWARDS, 1996). A definição de um modelo de política de colaboração e controle para a interação robô-humano deve prever o relacionamento entre os agentes humanos e robóticos em um mesmo ambiente, onde a tarefa colaborativa ocorrerá. A pesquisa nessa área tem obtido excelentes progressos na definição de estruturas e ontologias, porém a sua aplicação no controle de robôs ainda é novidade.

⁸ <http://www.w3.org/TR/owl-features/>

⁹ <http://www.w3.org/>

Existem duas formas distintas de se definirem políticas em sistemas, intra-regras e extra-regras. Na primeira, a política é definida nas próprias regras que podem referenciar outras regras. Na segunda, a definição da política é feita externamente.

Em Tonti et al. (2003) é apresentado um estudo comparativo entre três linguagens propostas para a representação de políticas de interação na Web Semântica. A Tabela 1 apresenta a comparação das principais características dessas linguagens em relação ao modelo M-Forum (MARTINS; CAMOLESI Jr, 2008). Neste trabalho, o modelo M-Forum foi adotado para a representação das regras do ambientes colaborativo e será descrito em detalhes nas próximas seções.

Tabela 1 – Comparação entre M-Forum e outros modelos de regras

	Kaos	Rei	Ponder	M-Forum
Suporte a ontologia	Sim	Sim	Não	Sim
Linguagem especificada	DAML/OWL	Baseada em Prolog	Linguagem Ponder	L-Forum
Ferramenta de especificação da política	KPAT – editor gráfico	Não (GUI em desenvolvimento)	Editor gráfico	Não (GUI em desenvolvimento)
Suporte a raciocínio	Java Theorem Prover	Prolog engine; Modelo evento-condição-ação	Cálculo de evento	Teorema da atividade; Teorema deôntica; Modelo evento-condição-ação.
Mecanismo de aplicação	Automação da política será explorada na próxima versão	Execução da ação não é prevista pela Rei engine	Interfaces Java para agentes de aplicação	Execução da regra não é prevista pela engine
Flexibilidade	Ontologia pode ser estendida com descrições dependentes de domínio para as entidades locais	Ontologia pode ser estendida com descrições dependentes de domínio para as entidades locais	Gerenciamento do domínio consiste técnica estruturada para partição de objetos complexos	Ontologia pode ser estendida com descrições dependentes de domínio para as entidades locais
Elementos representados	Ator, ação, grupo e lugar	Sujeito, atividade e objeto	Sujeito, atividade, objeto e domínio	Ator, atividade, objeto, tempo, espaço, associação, domínio, e abstrações de composição e generalização

4.3.4.1. O modelo M-Forum

O M-Forum (MARTINS; CAMOLESI Jr, 2008) é um modelo adequado à definição de regras e políticas para situações que envolvem a colaboração de diferentes tipos de agentes. O modelo permite a modelagem de ambientes colaborativos e fornece suporte à abstração de cinco dimensões distintas: ator; atividade; objeto; tempo e espaço. A dimensão ator permite a representação de um agente em um ambiente colaborativo, seus direitos, proibições e obrigações. Conforme mostra o Quadro 1, cada ator humano tem um identificador (Ach_id), um estado atual ($AchState$) e um conjunto de atributos (Ach_AttS). Atores não-humanos (Acs) possuem representação semelhante. Dado qh como o número de atores humanos no ambiente e qs , o número de não-humanos.

$$\begin{aligned} AchS &= \{Ach_1, Ach_2, \dots, Ach_{qh}\}, AcsS = \{Acs_1, Acs_2, \dots, Acs_{qs}\} \\ AchS &\neq \emptyset \vee AcsS \neq \emptyset \\ AchS \cap AcsS &= \emptyset \\ AcSS &= AchS \cup AcsS \\ Ach_i &= (Ach_id_i, AchState_i, Ach_AttS_i) \\ Acs_i &= (Acs_id_i, AcsState_i, Acs_AttS_i) \end{aligned}$$

Quadro 1 – Formalização da dimensão ator do M-Forum

Atores são responsáveis pela execução de atividades individuais ou coletivas, que podem afetar objetos, um ator ou um grupo de atores. Atividade é uma unidade de execução que pode ser realizada por um ator ou grupo. Geralmente, as atividades envolvem a manipulação ou a transformação de um objeto. Atividades devem ser definidas por operadores de Atividade e Lógicos, que representam direitos, proibições e obrigações. Os operadores de Atividade são necessários para especificar a interação entre atores e objetos. Atividades têm identificadores (At_id), um estado ($AtState$), um subconjunto de atividades (At_S), um subconjunto de operações (OpS) e um conjunto de atributos ($At_AttributeS$). Dado qa como o número de atividades de um ambiente, a dimensão atividade é formalmente apresentada no Quadro 2.

$$\begin{aligned} AtSS &= \{At_1, At_2, \dots, At_{qa}\} \\ At_i &= (At_id_i, AtState_i, AtS_i, OpS_i, At_AttributeS_i) \\ AtS_i &\subseteq AtSS \\ OpS_i &\subseteq OpS \end{aligned}$$

Quadro 2 – Formalização da dimensão atividade do M-Forum

Objetos representam elementos que constituem conceitos ou entidades do mundo real. Um objeto descreve-se pela representação de suas características estruturais e de seu comportamento. Atividades podem afetar ser realizadas sobre objetos e alterar suas características e seus estados. A modelagem de um objeto estabelece a uniformidade de visão e de tratamento, o que é útil para projetos de ambientes colaborativos. Como apresentado no Quadro 3, um objeto pode ser composto por outros objetos (*CompObS*) e possui um identificador (*Ob_id*), um estado e um conjunto de atributos (*Ob_AttS*). Atividades e operações podem ser realizadas sobre objetos, o que permite a modificação de seus estados e atributos.

$$\begin{aligned} \text{ObSS} &= \{\text{Ob}_1, \text{Ob}_2, \dots, \text{Ob}_{qo}\} \vee \text{ObSS} \neq \emptyset \\ \text{Ob}_i &= (\text{Ob_id}_i, \text{CompObS}_i, \text{ObState}_i, \text{Ob_AttS}_i) \\ \text{CompObS}_i &\subseteq \text{ObSS} \end{aligned}$$

Quadro 3 – Formalização da dimensão objeto do M-Forum

Espaço define a localização de atores ou objetos e as áreas específicas onde atividades e operações podem ocorrer. Da mesma forma que os demais elementos apresentados, espaços são essenciais para a modelagem de ambientes colaborativos.

Em uma interação colaborativa, elementos da dimensão espaço devem ser definidos por um operador de Espaço (*SpOp*) que especifica a posição ou o tamanho de atores e objetos em ambientes colaborativos. O elemento espaço possui um identificador (*Sp_id*), um estado (*SpState*) e um conjunto de atributos (*SpAttS*). Dado *qe* como o número de espaços em um ambiente, complementa-se a formalização da dimensão espaço conforme mostra o Quadro 4.

$$\begin{aligned} \text{Sp}_i &= (\text{Sp_id}_i, \text{SpState}_i, \text{SpAttS}_i) \\ \text{SpSS} &= \{\text{Sp}_1, \text{Sp}_2, \dots, \text{Sp}_{qe}\} \\ \text{SpOp} \in \text{So} &= \{ <, <=, >, >=, =, <>, ==(\text{attribution}), \text{not equal}, \\ &\text{inside, outside, intersect, meet, overlap, north, south, east, west} \} \end{aligned}$$

Quadro 4 – Formalização da dimensão espaço do M-Forum

Na modelagem do tempo, Duração, Data e Ocorrência representam a semântica básica para o estabelecimento de referências temporais. Essas semânticas são utilizadas na definição da lógica temporal de operadores para duração, data de ocorrência ou intervalo de ocorrência das atividades definidas nas interações entre atores e objetos.

A formalização básica para elementos temporais baseia-se no conjunto de números naturais (N), e representações para anos (Ty), meses (Tm), dias (Td), horas (Th), minutos (Tmi) e segundos (Ts), para um Momento ou Intervalo. Conjuntos enumerados de valores relativos (Tmr , Tdr , Thr , $Tmir$, Tsr) são utilizados na representação de datas de um calendário específico. Dado qt como o número de momentos ou intervalos de tempo que ocorrem em um ambiente, então se descreve a dimensão tempo como apresenta o Quadro 5

$Ty, Tm, Td, Th, Tmi, Ts \in N$
 $Tmr \in \{1, 2, 3, 4, \dots, 12\}$
 $Tdr \in \{1, 2, 3, 4, 5, \dots, 31\}$
 $Thr \in \{0, 1, 2, 3, \dots, 23\}$
 $Tmir \in \{0, 1, 2, 3, \dots, 59\}$
 $Tsr \in \{0, 1, 2, 3, \dots, 59\}$
 $Time = \{Ti_id, (Ty, Tm, Td, Th, Tmi, Ts)\}$
 $Datetime = \{Ti_id, (Ty, Tmr, Tdr, Thr, Tmir, Tsr)\}$
 $Tb, Te \in Datetime, Interval = \{Ti_id, (Tb, Te)\}$
 $TiSS = Time \cup Datetime \cup Interval$
 $TiSS = \{Ti_1, Ti_2, \dots, Ti_{qt}\}$
 $TiOp \in To = \{<, <=, >, >=, =, <>, == \text{ (attribution)},$
 precedes, succeeds, directly precedes, directly succeeds,
 overlaps, is overlapped by, occurs during, contains, starts,
 is stated with, finishes, is finished with, coincides with }

Quadro 5 – Formalização da dimensão tempo do M-Forum

4.3.4.2. A sintaxe L-Forum

L-Forum é uma linguagem desenvolvida para a formalização dos conceitos especificados pelo modelo M-Forum. Ela permite a definição precisa de regras que descrevem a interatividade em ambientes colaborativos. A estrutura geral de uma regra, em L-Forum, é definida pelo uso de três conjuntos distintos de cláusulas:

- Contexto: composto por parâmetros para a execução ou ativação de uma regra e pelas condições de sua aplicabilidade que estabelecem os cenários (valores de atributos, aspectos temporais ou espaciais);
- Definição (corpo): conjunto de expressões que estabelecem as ações ou condições para interações entre elementos, e envolvem aspectos transitórios de tempo e espaço. Regras e ações podem ser invocadas no corpo e permitem seqüenciar expressões complexas ou simples;

- Regime: conjunto opcional regras inter-relacionadas que tenham a mesma orientação para serem executadas ou aplicadas. Também podem ser definidos os cenários (valores de atributos, aspectos temporais ou espaciais) para ativação/desativação de uma regra.

Os elementos principais da sintaxe L-Forum são apresentados na Tabela 2.

Tabela 2 – A sintaxe completa L-Forum¹⁰

<rule> ::=	'Rule' <rule name> '[' <status> ']' '{' [<context>] 'Body' :: '<definition> [<regime>] '}'
<context> ::=	['Parameters: (' <parameters> ')'] [<applicability>]
<definition> ::=	<condition> <action> <rule call> [<definition>]
<regime> ::=	<survivability> ['Priorities: ' <priority>]
<parameters> ::=	<identifier> ':' <element> [',' <parameters>]
<element> ::=	<actor> <group> <object> <space> <time> <association> <activity> <operation>
<type> ::=	'actor' 'group' 'object' 'space' 'time' 'association' 'activity' 'operation'
<applicability> ::=	'Applicability: ' <condition expression>
<survivability> ::=	'Survivability: ' <condition expression>
<condition> ::=	'If ' <condition expression> 'then { ' <definition> ' }' ['else { <definition> ' }]
<action> ::=	'Action: (' <supreme action> <definition action> <attribution action> (<actor> <normative operator> {<activity> (<actor> <object>)} [<space attribution operation> <space>] [<time attribution operation> <time>]) [' ; ' <action>] ' ; '
<supreme action> ::=	<actor> <normative operator> <primitive operator> (<element> <domain> ('is part of' 'is a') <element>) <actor> <normative operator> <primitive group operator> <group> <element>)
<definition action> ::=	<actor> 'set' <status>
<attribution action> ::=	<actor> 'attribute' (<value> <formula> ((next prior) (<value domain> <domain name>)) <attribute>
<condition expression> ::=	'(' (<attribute><attribute condition operator>(<value> (['all' 'any'] (<value domain> <domain name>))) ((<element> <rule name>) <status condition operator> <group>) <value>) (<element>) <element group operator> (<element>) <group group operator> <group>) ([not] 'exist' <association>) (<element> <space condition operator> (<space> (<group> <domain name>))) (<attribute><space condition operator>(<value> 'here' (['all' 'any'] (<value domain> <domain name>)))) (<attribute><time condition operator>(<value> 'now' (['all' 'any'] (<value domain> <domain name>)))) (<actor> <activity condition operator> <activity>) (<condition expression> (and or)) ' '
<rule call> ::=	'Rule (' <rule name> ' (' <parameters> ') ' <normative operator> ['] ; ' <rule call>] ' ; '
<priority> ::=	<name> [' , ' <priority>]
<group> ::=	[(<identifier>) 'any' 'all' <integer value>] [<association>] [' : ' <name>] [' : Group ']

¹⁰ Fonte: Martins e Camolesi Jr (2008)

<actor> ::=	[(<identifier>) 'any' 'all' <integer value>] [<association>] [<group> '.'] [':<name>'] [':Actor']
<activity> ::=	[<activity> '.'] <name> [':Activity']
<operation> ::=	<activity> '.<name>' [':Operation']
<object> ::=	[(<identifier>) 'any' 'all' <integer value>] [<association>] [<object> '.'] [':<name>'] [':Object']
<space> ::=	[(<identifier>) 'any' 'all' <integer value>] [<association>] [<space> '.'] [':<name>'] [':Space']
<time> ::=	[(<identifier>) 'any' 'all' <integer value>] [<association>] [':<name>'] [':Time']
<association> ::=	<element> '.<name>' [':<association>'] [':Association']
<attribute> ::=	<element> '.<name>' [':Attribute']
<domain> ::=	<name> (<value domain> <grouping>)
<value domain> ::=	'(' (<numeric value> {',' <numeric value>}) (<string> {',' <string>}) ')'
<grouping> ::=	(<type> <name> <attribute condition operator> (<value> (['all' 'any'](<value domain> <domain name>)))&{and or} <grouping>) (<element> {',' <element>})
<status> ::=	<element> <status attribution operator> <value>
<primitive group operator> ::=	'insert' 'delete' 'update'
<primitive operator> ::=	'create' 'destroy'
<group element operator> ::=	'∈' '∉'
<group group operator> ::=	'⊆' '⊄' '⊂'
<activity condition operator> ::=	['not'] 'has'
<normative operator> ::=	'right' 'prohibition' 'obligation' 'dispensation'
<activity attribution operator> ::=	'receive'
<status attribution operator> ::=	'put on' 'move to'
<space attribution operator> ::=	'==' 'inside' 'outside' 'north' 'south' 'east' 'west'
<time attribution operator> ::=	'in' 'on' 'at'
<attribute condition operator> ::=	'<' '<=' '>' '>=' '=' '<>'
<time condition operator> ::=	'<' '<=' '>' '>=' '=' '<>' 'precedes' 'succeeds' 'directly precedes' 'directly succeeds' 'overlaps' 'is overlapped by' 'occurs during' 'contains' 'starts' 'is stated with' 'finishes' 'is finished with' 'coincides with'
<space condition operator> ::=	'<' '<=' '>' '>=' '=' '<>' 'not equal' 'inside' 'outside' 'intersect' 'meet' 'overlap' 'north' 'south' 'east' 'west'
<arithmetic operator> ::=	'+' '-' '/' '*'
<status condition operator> ::=	['not'] 'is'
<rule name> ::=	<string>
<name> ::=	<string>
<domain name> ::=	<string>
<string> ::=	<letter>,{<letter>}
<integer value> ::=	<digit>,{<digit>}
<real value> ::=	<digit>,{<digit>},'.',<digit>,{<digit>}
<numeric value> ::=	<integer value> <real value>
<digit> ::=	'1' '2' '3' '4' '5' '6' '7' '8' '9' '0'
<letter> ::=	'a' 'b' 'c' 'd' 'e' 'f' 'g' 'h' 'i' 'j' 'k' 'l' 'm' 'n' 'o' 'p' 'q' 'r' 's' 't' 'u' 'v' 'x' 'w' 'y' 'z'
<identifier> ::=	<letter> <identifier> [,<letter> <integer value>]
<formula> ::=	'('(<attribute> <numeric value> <formula>) [<arithmetic operator> <formula> '']
<value> ::=	<string> <numeric value>

Um exemplo de aplicação da estrutura sintática da L-Forum, obtido de Martins e Camolesi Jr (2008), é apresentado na Tabela 3, e descreve a regra “Julgar Artigo” com tempo de vida até o ano de 2010 (*survivability*). No exemplo, a palavra *now* indica a data atual e *A.possui.V.resultado* indica o atributo “resultado” da avaliação V, que é vinculada ao artigo A pela associação “possui”. *Set* é um operador para atribuição de estado, que no caso, pode atribuir “aceito” ou “rejeitado”.

Tabela 3 – Exemplo de regra em L-Forum

Rule Julgar Artigo [active] {			
Parameters:: (Au: Autor, A:Artigo, V:Avaliação, C:Comitê_de_programa)			
Applicability:: (now <= 01/09)			
Body:: If (A.possui.V.resultado = bom) or (A.possui.V.resultado = ótimo)			
then {	Action (C set aceito A);		
	Rule (Avisar Aceitação (C, Au, A, V), obligation);	}	
else {	Action (C set rejeitado A);		
	Rule (Avisar Rejeição (C, Au, A, V), obligation);	}	
Survivability:: (now <= 2010)			
}			

4.4. CONSIDERAÇÕES FINAIS

Como abordados neste capítulo, os ambientes colaborativos consistem modelos complexos, que incluem atores e objetos. Os atores são responsáveis por grande parte das atividades que alteram o estado do ambiente onde estão inseridos. Essas atividades são qualificadas por propriedades temporais e espaciais.

Agentes de *software* podem representar a interação de atores (humanos ou não) em um ambiente colaborativo, mas, para tanto, devem compartilhar uma representação comum do ambiente. Essa representação pode ser abstraída em modelos simbólicos adequados, como em ontologias e em políticas.

O modelo M-Forum permite abstrair cinco dimensões distintas de um ambiente colaborativo: atores, atividades, objetos, tempo e espaço. Sua linguagem, L-Forum, possui regras de formação bem definidas e não ambíguas, que permitem sua interpretação por seres humanos e por sistemas computadorizados. Sendo assim, tal modelo foi adotado neste trabalho para a representação das regras de colaboração, compartilhadas entre agentes humanos e robóticos.

5 COOPERAÇÃO ENTRE PARTES DE UM MANIPULADOR

Como apresentado no capítulo 2, o uso cooperativo de robôs encontra justificativa em diversas situações, como: limitações físicas do envelope de trabalho do manipulador para alcançar todas as peças ou objetos; transporte de objetos muito grandes ou pesados; otimização de tempo de execução. Porém, quando se tratam de inovações no modelo de controle para robôs industriais, não são comuns na literatura arquiteturas que permitam a cooperação entre as diferentes partes de um robô manipulador. Essa abordagem pode permitir maior adaptação do robô a tarefas sensíveis a alterações do estado do ambiente, fato plausível quando se trata da interação com seres humanos.

Também se pode constatar que os modelos de controle disponíveis na literatura não atendem plenamente a integração das regras cooperativas e colaborativas, tema principal deste trabalho. Sendo assim, um novo modelo de controle distribuído, que permita a deliberação baseada em tais regras, é proposto e apresentado neste capítulo. Além disso, o modelo permite a representação das partes individualmente atuadas de um robô.

O modelo em camadas proposto prevê dois níveis distintos, um global e outro local. No nível local encontram-se as representações dos agentes embarcados no *target* (no controlador do robô), que apresentam características mais reativas, dadas as restrições de tempo de suas tarefas. No nível global, são definidos os agentes que descrevem o modelo virtual do ambiente e os agentes que deliberam os comportamentos cooperativo e colaborativo do robô.

O estudo baseou-se em um robô manipulador IBM SCARA 7545, cujo controlador foi redefinido (AROCA; TAVARES; CAURIN, 2007). Para se provar a viabilidade do modelo, e também a comunicação entre os agentes globais e locais, foi desenvolvido um simulador gráfico, Scara3D. A interface foi testada em teleoperação, através da rede local do Laboratório de Mecatrônica (EESC/USP), e os tempos de atraso foram medidos e analisados.

O simulador também foi testado através de um túnel de VPN (*Virtual Private Network*) pela Internet, habilitado entre o *target* (no Laboratório) e um computador remoto em Fortaleza, durante o evento SAC/ACM 2008, onde foram apresentados os resultados deste capítulo (MARTINS Jr, CAMOLESI Jr; CAURIN, 2008).

5.1. ANÁLISE DO PROBLEMA

O estudo iniciou-se pela análise do robô, um IBM SCARA 7545 (Figura 9), cujo controlador foi redefinido por um projeto de *retrofitting* com plataforma aberta (AROCA; TAVARES; CAURIN, 2007). Todo o sistema foi desenvolvido com recursos e ferramentas de *software* de código aberto, como o sistema operacional Linux e uma interface de tempo real para o *kernel* – RTAI (*RealTime Application Interface*) (AROCA, 2008). O sistema já conta com acionamento de motores e controlador PID (*Proportional Integral Derivative*), além de leitura de sensores, embarcados em um computador industrial.

Figura 9 – Foto do robô IBM SCARA do Laboratório de Mecatrônica – EESC/USP

A arquitetura adotada segue o esquema *host-target*, no qual um computador (o *host*) é utilizado para desenvolvimento e compilação do *software* que é embarcado no PC industrial (*target*). Ambos os computadores executam o sistema operacional Linux, mas o *target* possui *kernel* acrescido de módulos e da interface RTAI. O *hardware* do PC industrial, montado em um *rack CompactPCI*, contém os seguintes componentes:

- Placas: Inova AMD K6, Acromag Carriers, National Instruments I/O;
- *Industry Packs* (IP): Tews 48 Digital I/O, Tews IP Quadrature, Tews DAC.

O IP (*Industry Pack*) de quadratura é utilizado na leitura de quatro encoders do robô SCARA. Outro IP é utilizado para a conversão analógico/PWM (*Pulse Width Modulation*). Os módulos de *software* de tempo real embarcados permitem a implementação de diversos comandos para acionamento dos motores, além do acesso aos dados dos sensores. Tal acesso “baixo nível” ao robô permite, portanto, a aplicação de diferentes estratégias de controle.

Em Aroca, Tavares e Caurin (2007) são apresentados os resultados dos testes aplicados no sistema, em ciclos de PID com intervalos de 1ms. A latência do PC industrial apresentou média de 2 μ s, com pico de 14 μ s no pior caso. O tempo médio de comunicação entre os espaços de usuário e de *kernel* foi de 12ms, com pico de 21ms no pior caso. O uso de recursos da máquina teve média de 80% de processamento e total de 8MB de ocupação de memória, de 64MB disponíveis.

A fim de se permitir a adaptação do robô a tarefas distintas e sua reconfiguração, encontra-se também em desenvolvimento uma arquitetura de controle para o robô manipulador (TAVARES; AROCA; CAURIN, 2007). Tal arquitetura utiliza, como base, o OROCOS (*Open Robot Control Software*), um framework de código aberto para controle em tempo real de robôs e outras máquinas. Além de componentes para controle, como recursos para operação de sistemas de tempo real, ferramentas para cálculo de cinemática e dinâmica, e mecanismos de inferência, o OROCOS dispõe de facilidades para compilação cruzada entre plataformas, integração *host-target* e recursos para implementação em sistemas distribuídos (COULOURIS; DOLLIMORE; KINDBERG, 2007), como o CORBA (*Common Object Request Broker Architecture*).

O principal objetivo de tal trabalho é permitir que um robô, ao ser inserido em um novo ambiente possa reconfigurar-se facilmente para atender as exigências das tarefas que deverá executar. Ou seja, com o uso da arquitetura em desenvolvimento será possível reprogramar o robô em tempo de execução, com pouco retardo. Além disso, a nova função a ser desempenhada pelo robô pode necessitar do uso de novos sensores ou de uma nova ferramenta (*end effector*). Como um possível resultado direto, o trabalho pode permitir a integração entre o robô SCARA e a garra antropomórfica Kanguera (CAURIN; ALBUQUERQUE; MIRANDOLA, 2004), mostrada na Figura 10, de forma que se permita a conexão direta da garra ao robô, como se fosse um novo módulo. Dessa forma, o novo conjunto poderá assumir um comportamento diferente, resultante da reconfiguração do sistema.

Figura 10 – Foto da garra Kanguera

Porém, não fica clara na solução proposta a separação entre os contextos global e local de controle. A princípio, a reconfiguração do sistema, quando da inserção de um novo módulo, deve cumprir com objetivos globais predefinidos. Tal estratégia, portanto, limita bastante a adaptabilidade do sistema a mudanças contínuas no ambiente.

Uma possível melhoria no modelo de controle é admitir que cada parte (*link*) do robô manipulador seja um módulo, e não somente o *end effector*. Dessa forma, pode-se considerar cada módulo como uma entidade distinta do sistema e analisar separadamente os papéis que deve cumprir, de acordo com o nível do controle, local ou global. Sendo assim, uma nova estratégia de controle cooperativo entre as diferentes partes do robô pode permitir sua adaptabilidade a diversos tipos de modificações do ambiente, e a qualquer tempo.

5.2. PROPOSTA DE UM NOVO MODELO DE ARQUITETURA

Não é comum a disponibilidade de um robô manipulador industrial com plataforma aberta de controle, que permita alterações e adaptações livres ao usuário. Esse tipo de robô é configurado e otimizado, geralmente pelo seu fabricante, para uma determinada aplicação. Portanto, toda operação do robô deve ser prevista de acordo com sua configuração de trabalho, e não permite adaptações a variações no ambiente, como a inserção de um obstáculo em seu espaço de trabalho.

O *retrofitting* do robô SCARA com a inclusão de controle baseado em plataforma aberta, realizado pelo Laboratório de Mecatrônica da EESC/USP, favorece o desenvolvimento de novas abordagens de controle. Outro ponto favorável a ser notado é que a atual implementação do sistema de controle do robô oferece facilidades extras de comunicação de dados em rede, o que permite a aplicação de arquiteturas de sistemas distribuídos. O *host* conectado ao PC industrial (*target*) permite a operação de serviço Web, o que habilita a troca de mensagens via HTTP (*Hyper Text Transfer Protocol*). O *framework* OROCOS, atualmente em implantação, dispõe de outros recursos de comunicação, como o suporte a CORBA.

Figura 11 – Sistema de controle do robô SCARA, com plataforma aberta

Uma visão geral do sistema de controle do robô SCARA do Laboratório de Mecatrônica é apresentada na Figura 11. Pode-se nela notar a caixa de controle do robô, que fornece sua alimentação elétrica, e onde é também montado o PC industrial Inova (o *target*) com as placas e *Industry Packs* (IP). O computador (*host*) é conectado à mesma rede local (interna) do PC industrial. Isso permite que o *target* faça a carga do sistema operacional, através da rede, obtendo sua imagem de um ponto do sistema de arquivos do *host*. O próprio sistema de arquivos do *target* é um ponto de montagem de um sistema de arquivos em rede, disponível no *host*, o que facilita o processo de implantação do *software* de controle.

Como mencionada anteriormente, uma possível melhoria ao sistema atual de controle, consiste da individualização das diferentes partes do robô SCARA. Dessa forma, as partes poderiam ser tratadas individualmente como diferentes módulos. Para tanto, cada parte deve ter capacidades individuais de movimentação (atuação).

Com base nisso, foi proposta a subdivisão do robô SCARA em conjuntos de peças funcionalmente associadas, ou seja, os diferentes *links* móveis do robô. A Figura 12 apresenta um desenho com a idéia da subdivisão do robô. Além da base (Figura 12-a), que é afixada ao chão, o braço robótico dispõe de três outras partes móveis (Figura 12-b e Figura 12-c), ou *links*. Os dois primeiros realizam movimento de rotação, já o terceiro, opera a translação e rotação.

Figura 12 – Decomposição de um robô SCARA em módulos

O modelo de controle a ser aplicado, deve considerar aspectos que podem alterar o comportamento de cada módulo, bem como de todo o robô. Tome-se, por exemplo, a interação entre um braço robótico e um ser humano. A tarefa poderia envolver o contato físico entre o agente humano e a ferramenta do robô (no caso de interfaces hápticas, p.e). A imprevisibilidade dos movimentos do ser humano constantemente alteraria as restrições de movimentação do manipulador em seu envelope, o que demandaria a reconfiguração de cada parte do robô.

Da mesma forma, pode-se considerar a situação de um objeto adentrar o espaço que um dos *links* deveria ocupar, durante a execução dos movimentos definidos pelo planejador de trajetórias. Para se evitar uma colisão, duas situações podem ser consideradas. A primeira, que uma simples alteração do controle local daquele *link* resolveria o problema. E uma segunda (mais provável), que a reconfiguração da trajetória deve atingir o controle de outros módulos também. Este problema ilustra a necessidade de aplicação de dois contextos distintos de controle, local e global, além de sugerir o uso de uma estratégia baseada em cooperação.

O modelo multicamadas de controle proposto neste trabalho é uma adaptação do apresentado por Brooks (1986). Nesta nova abordagem, mostrada na Figura 13, foram incluídos os comportamentos cooperativo e colaborativo. Além disso, as diferentes funções foram claramente agrupadas em dois diferentes contextos – local e global – o que permite serem implementadas em um sistema distribuído.

Figura 13 – Modelo multicamadas de controle cooperativo e colaborativo

A seguir, é apresentada a descrição de cada uma das camadas do modelo proposto neste trabalho:

- Controlador do motor: descreve a camada mais baixa do modelo, altamente dependente e acoplada ao *hardware*. Os agentes de *software* desenvolvidos para este nível são executados localmente (embarcado) no *target*. Os requisitos temporais do sistema são muito rigorosos, e requerem tempo de reposta da ordem de 1 a 5 milissegundos. A principal função desta camada consiste em emitir sinais para os *drivers* que atuam diretamente em cada motor. Os dados dos sensores (como *encoders*) acoplados aos motores realimentam a malha de controle, a fim de se garantir o correto desempenho dos atuadores;
- Escalonador de Tarefas: desempenha um importante papel no contexto local da arquitetura, pois oferece meios para o agendamento e coordenação dos processos que operam cada movimento individual das partes (*links*) do robô. A execução de cada movimento individual é parte da estratégia definida pela camada superior (planejador de trajetória) a fim de realizar o reposicionamento geral do robô no ambiente;
- Gerador de Trajetórias: elemento de mais alto nível do contexto local. Também impõe restrições rigorosas de tempo real, mas com período maior para término. Sua principal função consiste em definir a movimentação individual de cada motor, obtida da decomposição de uma trajetória desejada para o robô entre dois pontos (cinemática inversa);
- Gerador do Ambiente Virtual: camada mais baixa do contexto global e que deve implementar o modelo geral do ambiente onde o robô se localiza. Neste nível são considerados aspectos geométricos que permitam a análise do espaço do trabalho do robô. Dados de sensores visuais e de proximidade podem ser obtidos e comparados com o estado atual do modelo internamente representado pelo agente de *software*. Estratégias globais para deslocamento do robô no ambiente podem ser tratadas neste nível. Tais funções demandam maior poder computacional e, em contrapartida, permitem flexibilidade nos tempos de resposta. Uma das principais vantagens em se ter um modelo fiel do ambiente real, é a possibilidade de simulação de

comportamentos para o robô, em um ambiente virtual. Sendo assim, todo o desenvolvimento e teste de funcionalidades de alto nível para os comportamentos cooperativo e colaborativo podem, em primeira instância, ser feitos no ambiente simulado;

- Comportamento Cooperativo: esta camada deve ser implementada por agentes do contexto global baseados no modelo do ambiente e em regras predefinidas (banco de dados de regras, descrito a seguir). Pode-se assim notar ambas as capacidades de comportamento reativa e deliberativa, de tais agentes. A capacidade reativa vem da análise do ambiente, representado em seu modelo interno, enquanto que a deliberativa é resultante de decisões sobre as regras de cooperação. Como exemplo de um comportamento puramente cooperativo, pode-se vislumbrar as situações em que o humano deixa o ambiente (em um jogo, p.e, quando o humano deixa a partida antes do fim) e, nesses casos, as deliberações do sistema devem considerar apenas o robô e suas partes que o compõem;
- Comportamento Colaborativo: vale-se da mesma estratégia definida para a camada inferior (comportamento cooperativo). As regras de colaboração, previamente definidas, são armazenadas no banco de dados e permitem que os agentes analisem-nas perante o estado atual do modelo do ambiente. O comportamento colaborativo difere-se do puramente cooperativo ao prever situações que dependem da colaboração entre os agentes para o cumprimento de metas comuns.

5.2.1. Arquitetura distribuída, baseada em multi-agentes

No modelo multicamadas apresentado na seção anterior são claramente percebidos os contextos local e global. Parâmetros distintos de criticalidade e de rigorosidade para os agentes de cada contexto podem ser individualmente tratados dentro de suas respectivas camadas e permitem definir o grau de acoplamento com o robô (*target*). Essa abordagem permitiu a definição de uma arquitetura distribuída para o modelo de controle, conforme mostra a Figura 14.

As funções do contexto local descrevem processos altamente rigorosos quanto ao tempo de resposta. Da mesma forma, não exigem grandes capacidades de processamento e memória. Caracterizam-se, portanto, como agentes locais, fortemente acoplados (*tightly coupled*) ao *target*. Por outro lado, no contexto global, os processos são menos rigorosos em relação ao tempo de resposta, mas demandam maiores capacidades de processamento e memória. Tais características sugerem que os agentes responsáveis não sejam embarcados no *target*, mas colocados em computadores remotos, fracamente acoplados (*loosely coupled*).

Figura 14 – Arquitetura distribuída para o controle cooperativo e colaborativo

Os agentes locais interagem diretamente com o *target*, por meio de placas para comunicação com atuadores e sensores. Cada parte (*link*) do robô, com seus sensores, motores e mecanismos, pode ser tratada individualmente na arquitetura como se fosse um módulo distinto. O gerador de trajetórias é responsável pela integração dos movimentos individuais de cada módulo. A tarefa cooperativa pode então ser tratada como um conjunto de trajetórias e operações do manipulador em um nível mais alto da arquitetura. Essa é uma das principais vantagens da arquitetura proposta neste trabalho.

Sendo assim, agentes remotos, especialistas em cada módulo, podem interagir com os agentes locais para providenciar um comportamento adequado ao modelo global do ambiente. Tal interação pode ser apoiada por protocolos de comunicação de dados (como CORBA, ou HTTP). O comportamento cooperativo pode então ser atingido pela deliberação conjunta desses agentes remotos,

baseados em regras de cooperação e no modelo do ambiente. A cooperação está geralmente associada à execução de uma tarefa específica.

Em um nível mais alto de abstração, trata-se o comportamento colaborativo do robô. A colaboração pode ser observada quando um robô interage com um conjunto de tarefas que contribuam com a realização de metas e trabalhos comuns a outros agentes do ambiente, inclusive humanos. Da mesma forma, tal comportamento é deliberado por decisões de agentes sobre a análise de regras colaborativas perante o estado atual do ambiente.

As regras que norteiam a cooperação e a colaboração são mantidas em um banco de dados de regras, e podem ser acessadas pelos agentes globais, nos respectivos níveis de atuação. O banco de dados (*database*) de regras é um artefato importante para a arquitetura, e foi projetado para armazenar todas as regras de comportamento cooperativo e colaborativo. Tais regras são sentenças bem formadas, construídas de acordo com um modelo de regras para ambientes colaborativos, como o *M-Forum* (MARTINS; CAMOLESI Jr, 2008).

5.3. SCARA3D: SIMULADOR 3D PARA TESTE DA ARQUITETURA

Com o objetivo de experimentar o modelo teórico apresentado, foi implementada uma interface homem-robô (HRI – *Human Robot Interface*) gráfica, em um projeto de *software* denominado Scara3D (MARTINS Jr; CAMOLESI Jr; CAURIN, 2008). A interface dispõe de recursos gráficos 3D para a representação do ambiente virtual, que corresponde à camada mais baixa da parte global (Agentes Globais) da arquitetura. Sendo assim, consistiu principalmente de uma prova de conceito da arquitetura, e pela qual foi possível validar alguns de seus elementos.

5.3.1. Projeto e desenvolvimento do *software*

Um Diagrama de Casos de Uso do Simulador Scara3D é apresentado na Figura 15. Nele, são descritas as interações possíveis entre os atores humano e robô, e o sistema. A presença de uma seta (orientação) nas extremidades da associação indica a direção da ação, ou seja, quem recebe a interação.

Figura 15 – Diagrama de Casos de Uso do Simulador Scara3D

O caso de uso “Interage com o robô” representa as ações do Humano para comandar os movimentos do Robô SCARA, seja apenas em ambiente simulado, ou mesmo, se a interface estiver conectada aos agentes locais, também no robô real. A Tabela 4 contém o detalhamento desse caso de uso.

Tabela 4 – Detalhamento do caso de uso “Interage com o robô”

Caso de uso:	Interage com o robô
Atores:	Humano, Robô SCARA
Descrição:	Neste caso de uso o ator Humano pode interagir com o modelo virtual do robô.
Ativação:	Esta funcionalidade está disponível em ambas as circunstâncias: Cenário 1: a interface não está conectada ao robô; Cenário 2: a interface está conectada ao robô.
Extensões:	1. Rotaciona eixo 1; 2. Rotaciona eixo 2; 3. Translaciona eixo 3.
Pré-condições:	1. A interface gráfica do simulador deve estar operante. Apenas para o Cenário 2: 2. O caso de uso “Conecta o servidor” (estendido de “Aciona agentes locais”) deve ser realizado com êxito.
Pós-condições:	1. O Robô SCARA deverá estar na última posição solicitada para cada um de seus eixos.
Fluxo normal:	Cenário 1: a interface não está conectada ao robô. 1. O Humano fornece a posição desejada para o eixo em questão e aciona o botão respectivo; 2. Dependendo de qual eixo foi escolhido, a respectiva extensão do caso de uso é acionada; 3. O valor da posição desejada é traduzido para o respectivo valor da posição do sensor (<i>encoder</i>) do motor envolvido; 4. O movimento é então aplicado à representação visual no ambiente 3D.
Fluxo alternativo:	Cenário 2: a interface está conectada ao robô. Acrescenta ao Cenário 1: 5. A nova posição é informada, via rede, aos agentes locais e o movimento é também aplicado ao Robô SCARA.

Caso de uso:	Interage com o robô
Exceções:	<p>Humano informa um valor de posição inválido (vazio ou não numérico) em qualquer um dos casos de uso estendidos.</p> <ol style="list-style-type: none"> 1. A mensagem de erro “O formato do número está incorreto!” é exibida; 2. O movimento não é realizado. <p>Humano informa um valor de posição inválido (excede os limites do movimento) em qualquer um dos casos de uso estendidos.</p> <ol style="list-style-type: none"> 1. A mensagem “O valor informado está fora dos limites estabelecidos!” é exibida; 2. O movimento não é realizado.

O caso de uso “Aciona agentes locais” permite que o Humano conecte ou desconecte a interface ao robô. Caso seja conectada, os movimentos solicitados para cada um dos eixos são também comandados ao robô real.

Além disso, o caso de uso prevê uma funcionalidade adicional para calibração do robô, que solicita o procedimento de calibração aos agentes locais. O detalhamento deste caso de uso é apresentado na Tabela 5.

Tabela 5 – Detalhamento do caso de uso “Aciona agentes locais”

Caso de uso:	Aciona agentes locais
Atores:	Humano, Robô SCARA
Descrição:	Neste caso de uso o ator Humano conectar ou desconectar a interface aos agentes locais que acionam o robô real. Uma funcionalidade extra permite que seja solicitado o procedimento de calibração do robô.
Ativação:	Este caso de uso estará sempre disponível.
Extensões:	<ol style="list-style-type: none"> 1. Conecta o servidor; 2. Desconecta o servidor; 3. Solicita calibração.
Pré-condições:	<ol style="list-style-type: none"> 1. A interface gráfica do simulador deve estar operante. Apenas para as extensões “Desconecta o servidor” e “Solicita calibração”: 2. A extensão “Conecta o servidor” deve ser realizada com êxito.
Pós-condições:	<p>Para a extensão “Solicita calibração”:</p> <ol style="list-style-type: none"> 1. Após o procedimento, o Robô SCARA estará calibrado e cada um de seus eixos na posição inicial (zero).
Fluxo normal:	<p>Extensão “Conecta o servidor”:</p> <ol style="list-style-type: none"> 1. O ator Humano deve informar os endereços do servidor e de sua porta (para conexão TCP); 2. Na sequência, deve acionar o botão “Conectar”. <p>Extensão “Desconecta o servidor”:</p> <ol style="list-style-type: none"> 1. O ator Humano deve acionar o botão “Desconectar”. <p>Extensão “Solicita calibração”:</p> <ol style="list-style-type: none"> 1. O ator Humano deve acionar o botão “Calibrar”; 2. Ao final da calibração uma mensagem de confirmação é exibida ao Humano.
Exceções:	Extensão “Conecta o servidor”:

Caso de uso:	Aciona agentes locais
	<p>Humano informa valores para o endereço e porta que apontam para uma conexão indisponível.</p> <ol style="list-style-type: none"> 1. Após tentar a conexão, sem sucesso, o sistema deve apresentar a mensagem “Não foi possível estabelecer a conexão com o servidor!”; 2. A conexão não é estabelecida. <p>Humano informa um valor de porta inválido (vazio ou não numérico).</p> <ol style="list-style-type: none"> 1. A mensagem “O formato do valor para a porta está incorreto!” é exibida; 2. A conexão não é estabelecida.

Foram também previstas para o sistema facilidades para observação da representação gráfica 3D da cena onde o modelo virtual do robô está inserido. Tais funcionalidades foram concentradas no caso de uso “Inspeciona o ambiente virtual”, pelo qual o ator Humano pode modificar a perspectiva da cena, mudando sua orientação em sua visualização 3D.

O detalhamento desse caso de uso é apresentado na Tabela 6.

Tabela 6 – Detalhamento do caso de uso “Inspeciona o ambiente virtual”

Caso de uso:	Inspeciona o ambiente virtual
Atores:	Humano, Robô SCARA
Descrição:	Neste caso de uso o ator Humano pode interagir com a cena 3D virtual, modificando sua orientação no espaço. São permitidas alterações na rotação, translação e escala da cena, além de um atalho para retorno à perspectiva original da cena.
Ativação:	Este caso de uso estará sempre disponível.
Extensões:	<ol style="list-style-type: none"> 1. Rotaciona a cena; 2. Translaciona a cena; 3. Altera a escala da cena; 4. Retorna à perspectiva inicial.
Pré-condições:	1. A interface gráfica do simulador deve estar operante.
Pós-condições:	1. Após a interação do Humano, a cena será redimensionada ou movimentada de acordo com a extensão do caso de uso aplicada.

Caso de uso:	Inspeciona o ambiente virtual
Fluxo normal:	<p>Extensão “Rotaciona a cena”:</p> <ol style="list-style-type: none"> 1. O ator Humano aciona o botão esquerdo do mouse sobre algum ponto da cena 3D e o movimenta; 2. Toda a cena é rotacionada sobre o seu ponto central, de acordo com os movimentos realizados. <p>Extensão “Translaciona a cena”:</p> <ol style="list-style-type: none"> 1. O ator Humano aciona o botão direito do mouse sobre algum ponto da cena 3D e o movimenta; 2. Toda a cena é sobre translação de acordo com os movimentos realizados. <p>Extensão “Altera a escala da cena”:</p> <ol style="list-style-type: none"> 1. O ator Humano aciona o botão central do mouse sobre algum ponto da cena 3D e o movimenta; 2. Quando movimentado para cima, a cena reduz sua escala; quando movimentado para baixo, a escala aumenta. <p>Extensão “Altera a escala da cena”:</p> <ol style="list-style-type: none"> 1. O ator Humano aciona o botão “Perspectiva inicial”; 2. A orientação, posição e escala da cena retomam as configurações iniciais.

A solução de *software* foi definida de acordo com padrões de desenvolvimento dos *frameworks* gráficos *Swing* (extensão para interfaces gráficas do usuário, em Java) e *Java3D* (extensão Java para a manipulação de desenhos e cenários em três dimensões).

Para a concepção do modelo virtual do robô Scara foram consideradas as suas especificações técnicas, apresentadas na Tabela 7. Os dois eixos (*links*) principais, 1 e 2, realizam o posicionamento da extremidade do braço em uma área de trabalho que define um plano.

O terceiro eixo, onde é afixada a ferramenta (como uma garra, por exemplo), realiza movimento de translação e de rotação.

Tabela 7 – Especificações do robô IBM Scara 7545

Parâmetro	Valor
Graus de Liberdade	4
Rotação do <i>link</i> 1 - θ_1	$0^\circ - 200^\circ \pm 1^\circ$
Rotação do <i>link</i> 2 - θ_2	$0^\circ - 135^\circ \pm 1^\circ$
Rotação do <i>link</i> 3 - θ_3	$\pm 185^\circ \pm 0.5^\circ$
Translação do <i>link</i> 3	0 – 250 mm

Foram então projetadas as classes de objetos do sistema, conforme o Diagrama de Componentes da Figura 16. A classe *Scara* representa em três dimensões o robô IBM SCARA 7545, em um *TransformGroup*, que permite concentrar as especificações geométricas, de material, de iluminação e de comportamento do objeto 3D no cenário virtual.

Figura 16 – Diagrama de Componentes do Simulador Scara3D

Figura 17 – Principais classes do Simulador Scara3D

Na Figura 17 são apresentadas as principais classes que implementam o Simulador Scara3D. Um método privado da classe, *loadFromVRMLFiles*, foi definido para a carga das representações vetoriais de cada parte do robô, armazenadas individualmente em arquivos VRML (*Virtual Reality Modeling Language*). Os métodos *setInitialConfiguration* e *setFinalConfiguration* são responsáveis pelo correto posicionamento de cada parte e associação dos comportamentos (*behaviors*) de rotação ou translação, de acordo com as especificações de cada *link* do robô.

Métodos *rotateLinkX* foram definidos para cada *link* (onde $X=\{1,2,3\}$), com a finalidade de implementar o comportamento de rotação apropriado, de acordo com as faixas limite preestabelecidas. Um método *translateLink3* foi definido para implementar o movimento do eixo de translação do robô, respeitando-se os limites de distância estabelecidos pelo fabricante para tal deslocamento. Os comportamentos de rotação e de translação foram projetados nas classes *ScaraRotationInterpolator* e *ScaraTranslationInterpolator*, respectivamente. Tais classes estendem *Behavior* e implementam o método *processStimulus* que procede os movimentos.

O método *processStimulus* é imediatamente disparado quando se altera a posição final (no caso da translação, pelo uso do método *setFinalPosition*) ou o ângulo final (no caso da rotação, pelo método *setFinalAngle*) dos respectivos eixos, conseqüentemente às chamadas dos métodos *rotateLinkX* e *translateLink3* da classe *Scara*. A classe *ScaraTCPClient* foi projetada para permitir a comunicação da interface com o robô, por meio de *sockets* TCP/IP, em redes de computadores.

A classe *Scara3D*, do tipo *JFrame*, concentra os componentes da interface gráfica do usuário (GUI – *Graphical User Interface*), e os respectivos eventos para interação do usuário. Em um painel (*JPanel*) foi inserido o universo virtual, contendo a cena 3D e o robô. Botões também foram adicionados para permitir que o usuário dispare o reposicionamento (rotação e translação) dos *links* do robô, além de restabelecer a posição inicial de observação (perspectiva) da cena. Eventos de mouse também foram adicionados ao universo virtual para permitir as operações de alteração de escala, rotação e translação de todo o conjunto, na observação da cena 3D.

Figura 18 – Tela do Simulador Scara3D

A interface gráfica mostrada na Figura 18 exhibe os seguintes componentes:

- Um painel, onde foi inserida a cena 3D (universo virtual e o robô);
- Um botão (*Initial perspective*) que permite colocar a janela do observador da cena na posição inicial;
- Três botões (*Link1*, *Link2*, e *Link3*) que permitem operar individualmente a interpolação (rotação ou translação) de cada *link*;
- Dois botões (*Connect* e *Disconnect*) que permitem operar a interface em modos conectada (*online*) ou desconectada (*offline*) ao robô real;
- Um botão (*Calibrate*) que, quando conectada ao robô, a interface submete uma requisição para sua calibração.

5.3.2. Testes e resultados

Para ilustrar a utilização do simulador alguns valores foram atribuídos aos campos de texto, conforme ilustrados pela Figura 18. No lado esquerdo da tela encontram-se os campos com valores individuais para três movimentos distintos: rotação até a posição angular de 90° para o *link* 1; rotação até a posição angular de 45° para o *link* 2; e translação até a posição 100 mm para o *link* 3. No lado direito estão os dados para conexão com a aplicação servidora do *target*, via TCP/IP; caso o botão *Connect* seja acionado, uma conexão pode ser estabelecida com o servidor.

Figura 19 – Imagens do robô Scara – a) webcam; b) foto

Se a interface gráfica do simulador estiver conectada ao *target*, os movimentos disparados – representados pelos botões *Link1*, *Link2* ou *Link3* – serão realizados em ambos os ambientes (virtual e real). A posição final assumida pelo robô virtual após a realização dos movimentos é mostrada na Figura 18. Na Figura 19 são apresentadas duas ilustrações do robô real, ao final dos movimentos: a impressão de tela de uma página Web com a imagem de *webcam* (a) e também uma fotografia tirada do robô (b).

Os resultados dos testes realizados com o *link 1* são mostrados na Figura 20. Cada ponto representa a posição final (eixo vertical) dos *encoders* convertida em graus, e o intervalo de tempo correspondente (eixo horizontal) em milissegundos para atingi-la. Pode-se observar que ao se utilizar valores pequenos (10 e 20 ms) para o retardo as tarefas de rotação em ambiente simulado foram completadas antes que o robô real as concluísse. Este resultado aponta para a necessidade de se ajustar o modelo do ambiente virtual ao robô real.

Figura 20 – Resultados da atuação do eixo 1 do robô

Testes semelhantes foram aplicados ao *link 2* cujos resultados são apresentados na Figura 21. Pode-se notar que, mesmo com pequenos valores de retardo, a posição final foi atingida pelo robô real em tempo igual ou menor que o simulado, na maioria dos movimentos realizados. O melhor desempenho pode ser explicado se considerados os diferentes valores de inércia na atuação dos motores, ou seja, o *link 1* é mais pesado e mais longo que o *link 2*.

Figura 21 – Resultados da atuação do eixo 2 do robô

5.4. CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentado o problema da cooperação entre partes distintas de um robô manipulador. Para solucioná-lo, introduziu-se um novo modelo de controle multicamadas, implementado em arquitetura distribuída, e que permite a inclusão de comportamentos cooperativos e colaborativos.

A fim de se provar a aplicabilidade da arquitetura de controle, um simulador gráfico foi desenvolvido para implementar o gerador do ambiente virtual, previsto pelo modelo. O simulador foi então integrado a um robô industrial, cujo controle foi feito com ferramentas abertas, e no qual foram implantados os agentes do contexto local. A experimentação da interface gráfica permitiu validar a integração e a comunicação entre os contextos global e local da arquitetura.

A principal vantagem em se manter um modelo virtual muito próximo ao real é a possibilidade de simular aplicações e cenários, inicialmente no ambiente virtual e, em seguida, no mundo real.

Uma vez resolvido o problema do modelo de controle cooperativo, restam agora a definição e a aplicação do modelo de regras para representar a colaboração entre os robôs e seres humanos. Tal problema é abordado no próximo capítulo.

6 COLABORAÇÃO ROBÔ-HUMANO

Um dos maiores desafios da pesquisa em Robótica é a execução conjunta de tarefas por seres humanos e robôs em um mesmo espaço de trabalho (KEMP; EDSINGER; TORRES-JARA, 2007). Diversas soluções para a colaboração entre humanos e robôs foram apresentadas, mas em sua maioria abordam problemas de interação. Dentre os problemas relatados, encontram-se os de interação física direta, como o transporte conjunto de um objeto (GROTEN et al., 2010) e treinamento por interfaces hápticas (PARK; YOO; HOWARD, 2010), e os que não prevêm o contato físico, como na predição de movimentos humanos (NAJMAEI; KERMANI, 2010).

Além dos casos de interação, foram identificados alguns modelos de aprendizado colaborativo para robôs. Em Breazeal, Hoffman e Lockerd (2004) é apresentado um interessante resultado no aprendizado de tarefas por robôs, por meio de interação com seres humanos, mas em seu trabalho o comportamento deliberativo é mantido no ser humano, que ordena qual tarefa colaborativa deve ser realizada em determinado momento. Na mesma linha, em Park e Howard (2010) é apresentado um trabalho de aprendizado robótico das regras de um jogo para crianças, por meio da observação do comportamento humano ao manusear as peças do jogo. Porém, em ambos os casos quem assume o papel realmente deliberativo é o ser humano; para o robô resta o papel secundário de copiar o comportamento humano.

Uma possível melhoria para esse modelo de colaboração consiste de dotar o agente robótico de capacidades deliberativas, ou seja, permitir que ambos os agentes, humano e robótico, compartilhem uma mesma abstração do ambiente. Esta mudança de paradigma pode ser muito útil nas situações em que se deseja tutorar o ser humano, durante o seu processo de aprendizado. Dentre as aplicações diretas da abordagem encontra-se a interação com crianças e pessoas que apresentam dificuldades físicas ou mentais, com capacidades restritas para o entendimento das políticas que regem o ambiente naquele contexto. Nesses casos, o comportamento deliberativo do robô consiste de exemplo que pode ser percebido e imitado pelo agente humano.

O grande problema dessa abordagem advém justamente de sua abrangência. Isso porque é praticamente impossível representar todos os estados mentais de um ser humano, para as diversas situações vivenciadas em um ambiente. Sendo assim, uma solução possível é limitar o modelo adotado à representação de um problema específico, que consiste o domínio da aplicação colaborativa nesse ambiente.

Neste capítulo buscou-se definir um modelo de regras adequado à representação da colaboração entre robôs e humanos. Para testar a aplicabilidade do modelo, o estudo de caso da colaboração robô-humano em um jogo foi proposto e desenvolvido. Tal estudo foi publicado em Martins Jr, Camolesi Jr e Caurin (2010).

6.1. PROPOSTA DO MODELO DE REGRAS COLABORATIVAS

O principal desafio enfrentado neste ponto do trabalho consiste da definição de um modelo de regras que permita a representação da colaboração entre robôs e humanos. Como mencionado anteriormente, o modelo formal M-Forum foi adotado neste trabalho para tal propósito. Sendo assim, nesta seção descreve-se a como esse formalismo pode ser aplicado à concepção das regras.

Em linhas gerais, o ambiente colaborativo aqui estudado deve envolver a presença de agentes robóticos e humanos, que podem compartilhar objetos e espaços. Além disso, ambos os agentes podem deliberar e realizar atividades, que afetam objetos e outros agentes. Atividades são normalmente contextualizadas pela interação entre atores e objetos e qualificadas quanto ao tempo e espaço de suas ocorrências. Sendo assim, pode-se estabelecer a seguinte correlação entre os elementos identificados nesse tipo de ambiente colaborativo e as dimensões do modelo formal M-Forum:

- Atores: entidades deliberativas representadas por robôs e humanos;
- Objetos: objetos que serão utilizados pelos atores;
- Atividades: interações entre atores e ações realizadas sobre objetos;
- Espaço: delimitação do espaço de atuação dos atores no ambiente, bem como a localização dos atores e objetos;
- Tempo: qualificam os momentos de aplicabilidade de certas regras, bem como a duração das atividades no ambiente.

6.1.1. Exemplo de aplicação do modelo de regras

Para ilustrar a aplicação do modelo, bem como o mapeamento das regras com a sintaxe L-Forum, um exemplo foi estabelecido: a passagem de um objeto entre os agentes, humano e robô. Apesar de muito simples, o exemplo proposto é capaz de ilustrar a representação dos diferentes elementos envolvidos no ambiente. Sendo assim, os passos para sua análise e mapeamento são descritos a seguir.

Inicialmente, considera-se que, para o robô identificar as diferentes situações de colaboração envolvidas na tarefa, deve valer-se de algum sistema de reconhecimento visual. A interação entre o robô e o humano deve ser limitada a uma área comum, previamente definida (a superfície de uma bancada, p.e). Dessa forma, um código visual deve ser estabelecido, pelo qual toda vez que o robô identificar a mão aberta do ser humano na área comum, reconhecerá a situação “Humano solicita objeto” e poderá ativar essa regra. Por outro lado, se o ser humano apresentar na área comum a sua mão segurando um objeto, a situação “Humano entrega objeto” pode ser reconhecida e a respectiva regra ativada.

A seguir, pode-se observar que o ser humano é quem estabelece a causalidade das ações nesse ambiente. Ou seja, as ativações das regras pelo robô são conseqüências do estado do ambiente alterado pelo ser humano. A deliberação da ação pelo robô é tomada como efeito da comparação da causa (situação percebida) com a regra colaborativa adequada. A regra também descreve que atividade deve ser realizada pelo robô. Enfim, podem ser definidas as regras que definirão a representação dos elementos dimensionais abstraídos da tarefa colaborativa no ambiente. O conjunto total das regras, apresentado na Tabela 8, define a Política de Controle e Colaboração desse ambiente colaborativo.

Tabela 8 – Regras L-Forum para a passagem colaborativa de objetos

Rule	Humano Entrega Objeto [active] {
Parameters::	(hu: humano, ro: robô, obj: objeto, ac: área Comum)
Applicability::	(hu.mão not is aberta) and (ro.garra is aberta) and (hu.mão inside ac) and (ro.garra outside ac)
Body::	Rule Robô Move para Área (hu, ro , ac); Action (ro obligation set garra to fechada); Rule Robô Deixa a Área (hu, ro , ac)
	}

Rule	Humano Solicita Objeto [active]	{
Parameters::	(hu: humano, ro: robô, obj: objeto, ac: área Comum)	
Applicability::	(hu.mão is aberta) and (ro.garra not is aberta) and (hu.mão inside ac) and (ro.garra outside ac)	
Body::	Rule Robô Move para Área (hu, ro , ac); Action (ro obligation set garra to aberta); Rule Robô Deixa a Área (hu, ro , ac)	
	}	

Rule	Robô Move para Área [active]	{
Parameters::	(hu: humano, ro: robô, ac: área Comum)	
Applicability::	(hu.mão outside ac)	
Body::	Action (ro obligation set link1 move to inside ac); Action (ro obligation set link2 move to inside ac); Action (ro obligation set link3 move to inside ac)	
Survivability::	Priorities: Humano Entrega Objeto, Humano Solicita Objeto	
	}	

Rule	Robô Deixa a Área [active]	{
Parameters::	(hu: humano, ro: robô, ac: área Comum)	
Applicability::	(hu.mão outside ac)	
Body::	Action (ro obligation set link1 move to outside ac); Action (ro obligation set link2 move to outside ac); Action (ro obligation set link3 move to outside ac)	
Survivability::	Priorities: Humano Entrega Objeto, Humano Solicita Objeto	
	}	

É importante salientar que, por simplicidade, as regras foram definidas para o ator robótico, ou seja, as ações previstas para o ator humano não foram inseridas. As principais características de cada elemento dimensional representado pelas regras são descritas a seguir:

- Atores: os atores são robôs ou humanos. Nota-se que o robô é composto por partes distintas (ou *links*, no caso do exemplo 3) que podem ser individualmente representados em regras específicas, como atores diferentes, o que não foi mostrado no exemplo;
- Objetos: por simplicidade, este exemplo foi generalizado para qualquer tipo de objeto;
- Espaço: podem ser notados três diferentes espaços no problema: a área comum que será compartilhada pelos atores para a passagem

do objeto; o espaço privado de cada um dos dois atores, ou seja, onde se localizam o humano e o robô, respectivamente;

- Atividades: as atividades envolvidas, para ambos os atores, consistem em entregar e obter um objeto do parceiro. Cada situação é reconhecida pelo robô de acordo com o estado da mão do humano na área comum.

6.2. COLLABORATIVE TIC TAC TOE: UM ESTUDO DE CASO

Nesta seção é apresentado um estudo de caso, que envolve a interação entre um robô e um ser humano como oponentes em um “jogo da velha” (*Tic Tac Toe*). Esse tipo de jogo foi escolhido porque suas regras são muito simples de aprender, e permite a participação de pessoas com diferentes níveis de habilidade, de crianças a adultos. Também se levou em consideração o formato rígido do espaço de jogadas (*field*), o que permite a sua implementação em um tabuleiro, com locais predefinidos para as jogadas. Tais condições simplificam a coordenação dos movimentos realizados pelos oponentes no espaço comum delimitado pelo tabuleiro.

Tal estudo de caso foi proposto como uma prova de conceito do uso de regras colaborativas para governar as interações entre os dois diferentes tipos de atores, robô e humano. O estudo também se revela importante na demonstração da necessidade de definição de uma estratégia para seleção de regras em ambientes colaborativos, para que se possa compensar a imprevisibilidade de alguns comportamentos humanos.

6.2.1. Análise: regras do “jogo da velha”

O “jogo da velha” é disputado por dois jogadores no qual os participantes revezam-se no desenho de símbolos ('X' ou 'O') em uma matriz 3x3. Para vencer, um jogador deve desenhar três símbolos em uma linha, coluna ou diagonal da matriz. Quando a matriz estiver totalmente cheia e nenhuma seqüência de três símbolos iguais ocorrer (em uma linha, coluna ou diagonal), o jogo chegou a um empate. Uma possível situação ocorrida durante uma partida de “jogo da velha” é apresentada na Figura 22. Pode-se observar que o jogador que desenha símbolos 'X' venceu.

Figura 22 – Uma partida de “jogo da velha”

Por se tratar de um jogo simples, toda a estratégia do “jogo da velha” pode ser descrita por um conjunto de algumas regras (CROWLEY; SIEGLER, 1993), como apresentadas na Tabela 9. Tais regras são suficientes para se descrever todas as situações durante uma partida de “jogo da velha”. Porém, em algumas circunstâncias, pode ocorrer que mais de uma regra seja elegível, o que aponta para a necessidade de se definir critérios para a seleção da mais apropriada.

Tabela 9 – Conjunto de regras do “jogo da velha”

Jogada	Condições	Ação
Vencer	SE há uma linha, coluna ou diagonal com duas peças minhas e um espaço livre	Jogar no espaço livre
Bloquear	SE há uma linha, coluna ou diagonal com duas peças do meu oponente e um espaço livre	Jogar no espaço livre
Intersectar	SE há uma de minhas peças e dois espaços livres em duas linhas, colunas ou diagonais que se intersectam E SE a interseção está livre	Jogar na intersecção
Bloquear e intersectar (1)	SE há uma peça de meu oponente e dois espaços livres em duas linhas, colunas ou diagonais que se intersectam E SE a interseção está livre E SE existe um local livre que me permita obter “duas peças em uma linha”	Jogar no local livre
Bloquear e intersectar (2)	SE há uma peça de meu oponente e dois espaços livres em duas linhas, colunas ou diagonais que se intersectam E SE a interseção está livre E SE NÃO existe um local livre que me permita obter “duas peças em uma linha”	Jogar na intersecção
Jogar no centro	SE o centro está livre	Jogar no centro
Jogar no canto oposto	SE meu oponente jogou em um canto E SE o canto oposto está livre	Jogar no canto oposto
Jogar no canto vazio	SE há um canto vazio	Jogar no canto vazio
Jogar no lado vazio	SE há um lado vazio	Jogar no lado vazio

Se o critério adotado para a seleção das regras for o da precedência, e se considerada a disposição ordinal das regras na tabela, então o jogador jamais perderia uma partida. No pior caso, empatará.

Apesar da simplicidade das regras, selecioná-las depende de diversos fatores, como atenção, conhecimento, dentre outros. Tais fatores são claramente influenciados pela idade do jogador e, sendo assim, o sistema deve permitir a aplicação de estratégias adequadas aos diferentes níveis de habilidades apresentados por seu oponente. Do contrário, uma criança nunca seria capaz de vencê-lo, e ficaria aborrecida com o jogo.

6.2.2. Definição do conjunto de regras colaborativas

A próxima etapa envolve a tradução das regras para o formato L-Forum. De acordo com a sintaxe L-Forum, uma regra deve ser criada pelo uso de alguns elementos da linguagem. Toda regra deve ter um nome único e a declaração de seu *status*. Os parâmetros de uma regra especificam os elementos envolvidos, como atores, espaço e objetos. A aplicabilidade refere-se às condições que tornam uma regra selecionável. O corpo de uma regra descreve ações que devem ser realizadas.

Com base em tais critérios, foram elaboradas as regras para o ambiente colaborativo que descreve uma partida do jogo. Cada linha da Tabela 10 descreve uma das oito regras do conjunto apresentado na Tabela 9: Vencer; Bloquear; Intersectar, Jogar no Centro; Jogar no Canto Oposto; Jogar no Canto Vazio; Jogar no Lado Vazio; Bloquear e Intersectar. Esta última representa ambas as situações (1 e 2) da mesma regra, que foram mostradas na Tabela 9. As regras foram constituídas e denominadas de acordo com as condições definidas nas jogadas de mesmos nomes (Tabela 9).

Tabela 10 – Regras do “jogo da velha” mapeadas em L-Forum

```

Rule Vencer[active] {
  Parameters::    (robô:actor; tabuleiro:space; peças:object:group; partida:object)
  Applicability:: (partida not is encerrada)
  Body::
    if ((any tabuleiro.linha has 2 peças and peças ∈ robô) and
        (any tabuleiro.linha.posição is vazia))
    then { Action (robô right jogar peça inside tabuleiro.linha.posição); }
    else {
      if ((any tabuleiro.coluna has 2 peças and peças ∈ robô) and
          (any tabuleiro.coluna.posição is vazia))
      then { Action (robô right jogar peça inside tabuleiro.coluna.posição); }
      else {
        if ((any tabuleiro.diagonal has 2 peças and peças ∈ robô) and
            (any tabuleiro.diagonal.posição is vazia))
        then { Action (robô right jogar peça inside tabuleiro.diagonal.posição); }
      }
    }
  }
}

```

```

Rule Bloquear[active] {
  Parameters::    (robô:actor; tabuleiro:space; peças:object:group; partida:object)
  Applicability:: (partida not is encerrada)
  Body::
    if ((any tabuleiro.linha has 2 peças and peças ∉ robô) and
        (any tabuleiro.linha.posição is vazia))
    then { Action (robô right jogar peça inside tabuleiro.linha.posição); }
    else {
      if ((any tabuleiro.coluna has 2 peças and peças ∉ robô) and
          (any tabuleiro.coluna.posição is vazia))
      then { Action (robô right jogar peça inside tabuleiro.coluna.posição); }
      else {
        if ((any tabuleiro.diagonal has 2 peças and peças ∉ robô) and
            (any tabuleiro.diagonal.posição is vazia))
        then { Action (robô right jogar peça inside tabuleiro.diagonal.posição); }
      }
    }
  }
}

```

Rule Intersectar[active] {

Parameters:: (robô:actor; tabuleiro:space; peças:object:group; partida:object)

Applicability:: (partida not is encerrada)

Body::

if ((any posição is vazia and (posição ∈ tabuleiro.linha and (tabuleiro.linha has 2 posições and (posições is vazia) and tabuleiro.linha has 1 peça and (peça ∈ robô)) and (posição ∈ tabuleiro.coluna and (tabuleiro.coluna has 2 posições and (posições is vazia) and tabuleiro.coluna has 1 peça and (peça ∈ robô)))))) or

(any posição is vazia and (posição ∈ tabuleiro.linha and (tabuleiro.linha has 2 posições and (posições is vazia) and tabuleiro.linha has 1 peça and (peça ∈ robô)) and (posição ∈ tabuleiro.diagonal and (tabuleiro.diagonal has 2 posições and (posições is vazia) and tabuleiro.diagonal has 1 peça and (peça ∈ robô)))))) or

(any posição is vazia and (posição ∈ tabuleiro.coluna and (tabuleiro.coluna has 2 posições and (posições is vazia) and tabuleiro.coluna has 1 peça and (peça ∈ robô)) and (posição ∈ tabuleiro.diagonal and (tabuleiro.diagonal has 2 posições and (posições is vazia) and tabuleiro.diagonal has 1 peça and (peça ∈ robô)))))))

then { Action (robô right jogar peça inside posição); }

}

Rule Jogar no Centro[active] {

Parameters:: (robô:actor; tabuleiro:space; peças:object:group; partida:object)

Applicability:: (partida not is encerrada)

Body::

if (tabuleiro.centro is vazia)

then { Action (robô right jogar peça inside tabuleiro.centro); }

}

Rule Jogar no Canto Oposto[active] {

Parameters:: (robô:actor; tabuleiro:space; peças:object:group; partida:object)

Applicability:: (partida not is encerrada)

Body::

if (any tabuleiro.diagonal has 1 peça and (peça ≠ robô))

then {

if (peça is in tabuleiro.diagonal.canto_superior)

then { Action (robô right jogar peça inside tabuleiro.diagonal.canto_inferior); }

else { Action (robô right jogar peça inside tabuleiro.diagonal.canto_superior); }

}

}

Rule Jogar no Canto Vazio[active] {

Parameters:: (robô:actor; tabuleiro:space; peças:object:group; partida:object)

Applicability:: (partida not is encerrada)

Body::

```

    if (tabuleiro.canto_superior_direito is vazio)
    then { Action(robô right jogar peça inside tabuleiro.canto_superior_direito); }
    else {
        if (tabuleiro.canto_superior_esquerdo is vazio)
        then { Action(robô right jogar peça inside tabuleiro.canto_superior_esquerdo); }
        else {
            if (tabuleiro.canto_inferior_direito is vazio)
            then { Action(robô right jogar peça inside tabuleiro.canto_inferior_direito); }
            else {
                if (tabuleiro.canto_inferior_esquerdo is vazio)
                then { Action(robô right jogar peça inside tabuleiro.canto_inferior_esquerdo); }
            }
        }
    }
}

```

}

Rule Jogar no Lado Vazio[active] {

Parameters:: (robô:actor; tabuleiro:space; peças:object:group; partida:object)

Applicability:: (partida not is encerrada)

Body::

```

    if (tabuleiro.linha_superior.centro is vazio)
    then { Action(robô right jogar peça inside tabuleiro.linha_superior.centro); }
    else {
        if (tabuleiro.linha_inferior.centro is vazio)
        then { Action(robô right jogar peça inside tabuleiro.linha_inferior.centro); }
        else {
            if (tabuleiro.coluna_esquerda.centro is vazio)
            then { Action(robô right jogar peça inside tabuleiro.coluna_esquerda.centro); }
            else {
                if (tabuleiro.coluna_direita.centro is vazio)
                then { Action(robô right jogar peça inside tabuleiro.coluna_direita.centro); }
            }
        }
    }
}

```

}

Rule **Bloquear e Intersectar**[active] {

Parameters:: (robô:actor; tabuleiro:space; peças:object:group; partida:object)

Applicability:: (partida not is encerrada)

Body::

```

    if (any posição_intersecção is vazia and (posição_intersecção ∈ tabuleiro.linha and
(tabuleiro.linha has 2 posições and (posições is vazia) and tabuleiro.linha has 1 peça and (peça ≠
robô)) and (posição_intersecção ∈ tabuleiro.coluna and (tabuleiro.coluna has 2 posições and
(posições is vazia) and tabuleiro.coluna has 1 peça and (peça ≠ robô))))))
    then { if((any posição_livre in tabuleiro.linha or tabuleiro.coluna is vazia) and
(posição_livre ∈ tabuleiro.linha and (tabuleiro.linha has posição and (posição is vazia) and
tabuleiro.linha has peça and (peça ∈ robô))))
        then { Action (robô right jogar peça inside posição_livre); }
        else { Action (robô right jogar peça inside posição_intersecção); }
    }
    else { if (any posição_intersecção is vazia and (posição_intersecção ∈ tabuleiro.linha
and (tabuleiro.linha has 2 posições and (posições is vazia) and tabuleiro.linha has 1 peça and (peça
≠ robô)) and (posição_intersecção ∈ tabuleiro.diagonal and (tabuleiro.diagonal has 2 posições and
(posições is vazia) and tabuleiro.diagonal has 1 peça and (peça ≠ robô))))))
        then { if((any posição_livre in tabuleiro.linha or tabuleiro.diagonal is vazia) and
(posição_livre ∈ tabuleiro.linha and (tabuleiro.linha has posição and (posição is vazia) and
tabuleiro.linha has peça and (peça ∈ robô))))
            then { Action (robô right jogar peça inside posição_livre); }
            else { Action (robô right jogar peça inside posição_intersecção); }
        }
    else { if (any posição_intersecção is vazia and (posição_intersecção ∈ tabuleiro.coluna
and (tabuleiro.coluna has 2 posições and (posições is vazia) and tabuleiro.coluna has 1 peça and
(peça ≠ robô)) and (posição_intersecção ∈ tabuleiro.diagonal and (tabuleiro.diagonal has 2
posições and (posições is vazia) and tabuleiro.diagonal has 1 peça and (peça ≠ robô))))))
        then { if((any posição_livre in tabuleiro.coluna or tabuleiro.diagonal is vazia) and
(posição_livre ∈ tabuleiro.linha and (tabuleiro.linha has posição and (posição is vazia) and
tabuleiro.linha has peça and (peça ∈ robô))))
            then { Action (robô right jogar peça inside posição_livre); }
            else { Action (robô right jogar peça inside posição_intersecção); }
        }
    }
}

```

6.2.3. Projeto e desenvolvimento do *hardware*

Dado que o principal propósito deste projeto não envolve a pesquisa por métodos para o controle e posicionamento precisos, mas sim a concepção de um modelo para interação alto nível, foi adotada uma estratégia simplificada de operação do robô baseada em pontos fixos. Todas as 19 posições que representam locais válidos no jogo foram predefinidas e marcadas com pequenos círculos em um tabuleiro com 800x400 mm, como mostra a Figura 23. Dez dessas posições – cinco em cada lateral do espaço de jogadas – são utilizadas como repositório das peças.

Figura 23 – Estampa do tabuleiro do “jogo da velha”

Dezenove *reed switches* (sensores de presença magnéticos), como os mostrados na Figura 24, foram afixados no tampo de madeira do tabuleiro (Figura 25), um em cada posição dos lugares válidos para as peças. Esse tipo de sensor é acionado todas as vezes que um dos pólos magnéticos de um ímã se aproxima de uma de suas extremidades.

Figura 24 – *Reed switch* e seu encapsulamento

Os dezenove sensores foram conectados às entradas digitais de um circuito multiplexador. Esse circuito permite a leitura de até trinta e duas entradas digitais e permite a leitura de seus estados através de uma porta paralela comum (LPT) de um microcomputador. Na Figura 26 é apresentada uma foto da vista inferior do tabuleiro, onde podem ser identificadas a placa multiplexadora, a fonte de alimentação e as conexões com os sensores.

Figura 25 – Vista superior do tampo do tabuleiro

Figura 26 – Vista inferior do tabuleiro: placa, fonte e conexões

Um pequeno botão pulsante (*push button*) foi conectado à vigésima entrada digital da placa multiplexadora. A função desse botão é representar a detecção do humano na área compartilhada, e poderá ser convenientemente substituído no futuro por um sensor de presença mais adequado (barreira ótica ou câmera, p.e).

As peças foram projetadas de forma a permitir fácil manuseio pela garra do robô, e confeccionadas por processo de usinagem em polímero. A garra implantada no TCP (*Tool Center Point*) do robô IBM SCARA 7545 é do tipo pneumática, com dois dedos. Sendo assim, a melhor solução consistiu da colocação de uma pequena haste cilíndrica vertical sobre cada peça. Porém, as peças deviam ser detectadas pelo sensor magnético, quando colocadas sobre cada uma das posições do tabuleiro. A solução adotada envolveu a fixação de ímãs (reciclados de falantes de microcomputadores), um em cada peça. Metade das peças recebeu uma marcação 'X' e a outra metade uma marcação 'O', como mostradas na Figura 27.

Figura 27 – Peças magnéticas confeccionadas para o jogo

Na Figura 28 é apresentado um diagrama que representa as partes que compõem o sistema distribuído. Nele, identificam-se o robô IBM SCARA 7545 (cujo modelo de controle foi descrito no capítulo anterior) e o tabuleiro. A integração de todo o sistema ocorre através da rede local (Ethernet), pela qual três computadores se comunicam: controlador do robô, leitor dos sensores e host (com a aplicação).

Figura 28 – Esquema geral do sistema distribuído

Uma foto do ambiente completo é apresentada na Figura 29. Nela, pode-se observar: o robô IBM SCARA 7545 com a garra pneumática (1); o controlador, onde o PC industrial Inova (*target*) e os demais elementos do controle estão instalados (2); o servidor Linux que inicializa o Inova e fornece sua imagem de boot (3); o tabuleiro do jogo com as peças (4); o host remoto que opera o jogo (5); computador que faz a leitura da placa multiplexadora, via porta paralela (6).

Figura 29 – Foto do ambiente completo

Alguns pesquisadores, a exemplo de Brooks (2008) e Breazeal, Hoffman e Lockerd (2004), relatam que as pessoas costumam interagir mais naturalmente com robôs que possuem aparência agradável, com formas humanas ou de animais. Ao se considerar que o experimento desenvolvido consiste de um jogo simples, e que crianças podem facilmente jogá-lo, surgiu a necessidade de torná-lo mais lúdico e esconder um pouco o seu aspecto de máquina. Sendo assim, o robô SCARA foi vestido com uma fantasia de elefante, como mostrado na Figura 30. Nessa figura também são apresentados o tabuleiro e as peças do jogo.

Figura 30 – Robô vestido de elefante durante uma partida de “jogo da velha”

6.2.4. Projeto e desenvolvimento do *software*

Como ilustrado pela Figura 28, o sistema distribui-se em três computadores distintos e em cada um deles uma parte do *software* é implantada:

- Computador que controla o robô: como mencionado no capítulo anterior, o *software* nele implantado representa as camadas inferiores do modelo (agentes locais) e foi desenvolvido em outro projeto (AROCA; TAVARES; CAURIN, 2007);
- Computador que se comunica com o tabuleiro: neste computador foi instalado um programa que realiza leituras periódicas dos sensores do tabuleiro e envia uma mensagem UDP (*User Datagram Protocol*) *broadcast* para a rede local;
- Computador que hospeda a aplicação (jogo): na verdade, a aplicação que coleta os dados do ambiente (tabuleiro), opera as regras do jogo, implementa as interfaces e comanda o robô, pode ser instalada no mesmo computador que realiza a leitura dos sensores, porém optou-se por outro por questões de desempenho e praticidade.

Inicialmente é descrita a aplicação que realiza a leitura dos sensores. A aplicação foi codificada na linguagem C e consiste de um laço infinito que realiza leituras dos sensores, através da porta paralela, a cada 10 milissegundos. Caso algum sensor modifique o seu estado em relação à leitura anterior, os estados de todos os sensores (representados individualmente por bits individuais) são seqüenciados em uma palavra de 32 bits que é enviada em uma mensagem UDP *broadcast* para a rede local (porta de destino 8765). Dessa forma, a aplicação de destino precisa apenas realizar leituras nessa porta e saberá das alterações de estado do tabuleiro pela chegada das mensagens.

Para o desenvolvimento da aplicação que representa o jogo, foi considerada a interação entre o robô SCARA e seres humanos, que podem ser classificados como atores, de acordo com as especificações M-Forum. O robô SCARA é composto por *links* rotacionais e translacionais e possui uma garra pneumática, enquanto que o “jogo da velha” possui um tabuleiro e peças ('X' e 'O'); tais partes representadas individualmente como objetos no modelo Forum. A abstração desses elementos e os relacionamentos entre eles são representados no diagrama de classes do modelo conceitual mostrado na Figura 31. Os atributos e métodos das classes foram suprimidos para melhor visualização do diagrama.

No diagrama de classes da Figura 31 podem ser notados os principais grupos de classes desenvolvidos:

- Superclasses do modelo M-Forum (cor azul);
- Classes que implementam atores e objetos (cor laranja);
- Classes que representam os sensores e atuadores (cor verde);
- Classes que realizam a comunicação de mensagens (cor amarela).

As primeiras, as superclasses do modelo M-Forum, são as classes de mais alto nível no diagrama e descrevem o ambiente (*Environment*) e as dimensões que o compõem: objetos (*ForumObject*), atores (*ForumActor*), atividades (*ForumActivity*), espaço (*ForumSpace*), e tempo (*ForumTime*).

As classes que implementam atores e objetos estendem as superclasses *ForumActor* e *ForumObject*. Elas descrevem os dois atores envolvidos, o robô (*ScaraRobot*) e o ser humano (*Human*). O robô é composto por *links* e garra, que consistem objetos distintos. Cada link também possui pelo menos um servomotor que o movimenta. Dois outros tipos de objetos são previstos no ambiente: o tabuleiro (*TicTacToeBoard*) e as peças (*TicTacToePiece*).

As classes que representam sensores e atuadores são responsáveis por integrar o modelo à arquitetura de controle e permitem tanto a detecção de alterações no ambiente, como a atuação das partes do robô. Para a detecção da posição de referência de cada servomotor foi projetada a classe *Encoder*, que permite armazenar o estado atual da posição de cada junta. Já as alterações de estado dos objetos no ambiente, limitada neste protótipo à posição das peças no tabuleiro, foram representadas por objetos do tipo *PresenceSensor*, que são associados um para cada lugar a ser monitorado (no caso do tabuleiro, um para cada sensor magnético).

O último grupo de classes é responsável por toda a comunicação de mensagens e integração do sistema. A comunicação com os agentes externos foi implementada com o uso de *sockets*. Ela prevê duas situações distintas:

- Recepção das mensagens do tabuleiro: um *UDPMessageMonitor* usa um *UDPMessageReceiver* para receber as mensagens. Para cada mensagem que chega da rede, com o *status* de todos os sensores de presença, é criado um objeto *UDPMessageHandler* que realiza os devidos ajustes de *status* nos objetos do tipo *PresenceSensor*;
- Leitura de *encoders* e atuação dos motores: para a atualização objetos do tipo *Encoder*, um *EncodersMonitor* usa a instância de um *TCPMessageClient*, que submete mensagens TCP de consulta da posição atual dos *encoders* aos agentes locais (no *target*). A atuação dos *servomotores* é operada por um objeto *MotorActuator* que também se utiliza de uma instância de um *TCPMessageClient* que comanda o movimento do motor específico com de mensagens aos agentes locais.

A comunicação interna de mensagens do sistema foi implementada com a passagem de eventos, para o que se utilizou de um padrão de desenvolvimento chamado *Observer* (ou *Listener*, em Java). Por esse modelo, uma classe que deseja ser ouvinte de um evento, deve implementar uma interface que a complementa de tal habilidade e cadastrar-se junto ao objeto que dispara tais eventos. As interfaces projetadas no sistema foram *UDPMessageListener*, *PresenceListener*, *PositionListener*, *BoardStatusListener*, que habilitam a recepção de mensagens por eventos específicos, respectivamente *UDPMessageEvent*, *PresenceEvent*, *PositionEvent* e *BoardStatusEvent*.

Figura 31 – Diagrama de classes do modelo conceitual do “jogo da velha”

6.2.4.1. Implementação do conjunto de regras colaborativas

A próxima etapa envolveu o projeto e a integração das regras ao ambiente distribuído que compõe o jogo. As regras, detalhadas na seção 6.2.2, apresentam características comuns, como o estado que indica se a regra está ativa, o teste de sua aplicabilidade e o conjunto de parâmetros de entrada. Essas características foram generalizadas pela classe abstrata *TicTacToeRule*, como se apresenta na Figura 32.

A classe *TicTacToeRule* mantém uma variável booleana *active* para indicar o estado de atividade da regra. As regras do jogo devem expressar uma ordem de precedência (CROWLEY; SIEGLER, 1993) e, sendo assim, uma variável inteira *priority* foi incluída no projeto da classe para indicá-la. Um método *isApplicable* foi implementado na classe para testar se a regra está ativa, se o jogo está em progresso e se a regra se aplica àquele estado do jogo; este último teste é realizado pela chamada do método abstrato *getTargetPosition*, implementado por todas as classes que estendem *TicTacToeRule*. O construtor de *TicTacToeRule* recebe, como parâmetros, a referência do objeto chamador *TicTacToeGame* que possui todo o estado do jogo (referências do robô, do tabuleiro, das peças e do estado da partida), além do valor para a prioridade e do estado de atividade da regra.

Como mencionado, cada subtipo de *TicTacToeRule* deve implementar o método *getTargetPosition* para fornecer o comportamento específico daquela regra. Tal comportamento é descrito no corpo (*body*) de cada uma das oito regras detalhadas na Tabela 10. Uma classe *RulesSelector* foi implementada para carregar o conjunto de regras em seu construtor, selecionar as regras aplicáveis e definir a qual será aplicada (de acordo com a prioridade) por um método *selectRule* e finalmente aplicar a regra, pelo método *applyRule*.

Figura 32 – Visão geral das classes de regras do sistema

6.2.4.2. Integração do robô IBM SCARA 7545 ao sistema

Para o desenvolvimento da aplicação para o “jogo da velha” um dos problemas resolvidos consistiu da demarcação dos pontos das dezenove posições válidas no tabuleiro, mostradas na Figura 23. A estratégia adotada envolveu o posicionamento visual do TCP em cada local (técnica utilizada na programação “ponto a ponto” para robôs industriais) e, para tanto, foi desenvolvida uma interface gráfica (*ScaraCalibration*) que permitiu o comando individual de cada link, mediante comunicação com o *target*, via rede. A interface gráfica é mostrada na Figura 33.

O procedimento consistiu em inicialmente comandar a calibração do robô e, em seguida, posicionar o TCP em cada ponto do tabuleiro, operando individualmente cada link. A interface gráfica apresenta elementos deslizantes (*sliders*) que permitem variar gradativamente a posição dos *links* entre seus limites máximos. Os *links* 1 e 2 são operados em movimentos rotacionais enquanto que o link 3 em movimento linear (translação).

Os comandos que operam a calibração e o posicionamento de cada link são enviados ao *target* em mensagens TCP/IP. Para calibrar o robô, uma mensagem com o conteúdo “CALIBRATE” deve ser enviada. O comando de posicionamento de um link *n* é realizado pelo envio de uma mensagem “REFERENCIA *n valor*”, onde *valor* é um número que representa a quantidade de pontos do *encoder* a ser percorrida a partir da posição inicial (definida na calibração).

Tabela 11 – Descrição do protocolo de operação do *target*

Operação	<i>n</i>	<i>valor</i>	Posição relativa
Rotação do link 1	1	1 a 157001	0° a 180°
Rotação do link 2	2	1 a 57001	0° a 130°
Translação do link 3	3	1 a 87001	0 mm a 230 mm
Abre/fecha a garra	9	0 ou 1	Aberta ou fechada

Após os procedimentos de calibração e ajuste do robô, foram definidas as posições de interesse, apresentadas mais adiante, na Tabela 15. Essas posições consistem valores que compõem o estado inicial da instância da classe *ScaraRobot*.

Figura 33 – Interface gráfica para ajuste do posicionamento do robô

Para ilustrar a operação do robô Scara pelo envio de mensagens TCP/IP ao *target* tome-se, por exemplo, o posicionamento do TCP sobre a posição 15 do tabuleiro, como se apresenta na Figura 34. De acordo com os dados da Tabela 15, o link 1 deve ser levado até a posição angular indicada pelo valor 51462 (59°), e o link 2 para a posição angular 34639 (79°). Para tanto, duas mensagens TCP/IP devem ser enviadas ao *target* com os respectivos conteúdos (indicados entre aspas): “REFERENCIA 1 51462” e “REFERENCIA 2 34639”.

Figura 34 – TCP do robô sobre a posição 15 do tabuleiro

6.2.4.3. Integração do simulador Scara3D ao sistema

Para comprovar a utilidade do ambiente virtual, cujo desenvolvimento foi inicialmente descrito no capítulo 5, uma etapa complementar no desenvolvimento do sistema foi cumprida, que consistiu da integração do simulador ao jogo. A estratégia adotada consistiu da modificação do projeto Scara3D, anteriormente definido como uma interface do usuário para interação com usuários, para se tornar um novo *target* virtual. Para tanto, foi necessário alterar o comportamento da interface, que passou a operar como um servidor TCP/IP, de acordo com o mesmo protocolo de aplicação do *target* real, descrito na seção anterior.

Uma nova classe *Tabuleiro* foi desenvolvida para complementar o ambiente virtual. Tal classe consiste de um objeto visual 3D (um tampo) com dimensões de 800 x 400 mm, onde foi estampado o desenho original do tabuleiro (Figura 23). A partir de então o objeto, com a devida aplicação de escala, foi adequadamente posicionado e orientado na cena 3D, inspirado no modelo real.

Figura 35 – TCP do Scara3D sobre a posição 15 do tabuleiro

A integração do tabuleiro à cena 3D é mostrada na Figura 35. Na interface foi também inserida uma caixa de texto para exibição das mensagens TCP/IP recebidas pelo *target* virtual. A figura mostra um teste de posicionamento do robô, que recebeu as mensagens “REFERENCIA 1 51462” e “REFERENCIA 2 34639”. Ao se comparar o resultado obtido na Figura 35 com o anteriormente apresentado na Figura 34, pode-se notar que ambos os *targets*, tanto o real e como o virtual, foram deslocados para a mesma posição do tabuleiro. Tal resultado comprova a proximidade dos dois ambientes, anteriormente declarada no capítulo 5.

A modificação do projeto Scara3D é apresentada na Figura 36. Pode-se notar a nova classe *Tabuleiro*, além da troca da classe *ScaraTCPClient* pela *ScaraTCPServer* para atender o novo papel (servidor) da interface. Além disso, pode-se também observar a inclusão de uma nova classe *TicTacToeGameGUI*, que passou a consistir do ponto principal de integração do jogo.

Figura 36 – Integração do Scara3D ao sistema

A classe *TicTacToeGameGUI*, mostrada na Figura 37, é uma interface gráfica *Swing* que exibe o estado atual do tabuleiro do Jogo da Velha, além do resultado do processo deliberativo de escolha da regra. Tais informações são fornecidas pela classe *TicTacToeGame*, e adequadamente atualizadas na tela. As informações sobre o posicionamento das peças no tabuleiro, obtidas dos sensores de presença, são utilizadas para atualizar sua representação gráfica na tela, pela adequada colocação de símbolos ‘X’ e ‘O’ em cada marca. Quando o sistema seleciona uma regra, ela é realçada na interface e o resultado de sua aplicação (posições inicial e final do *pick-and-place* do robô) é também exibido.

Figura 37 – Interface gráfica do Jogo da Velha

Toda interação do humano e do robô deve ser realizada sobre o tabuleiro real, pela movimentação das peças, dado que a interface gráfica não dispõe de quaisquer componentes que permitam interferência no jogo. A única interação disponível é realizada pelos botões que exibem a interface virtual 3D. O botão *Scara3D* mostra a janela com o ambiente virtual; os botões *Geral*, *Superior*, *Tabuleiro* e *Frontal* modificam o ponto de observação da cena, nas respectivas perspectivas.

Foram também implementadas no sistema diversas condições de contorno, que permitem detectar se o jogo está em andamento, ou se está parado por vitória, empate ou violação de jogada. O jogo só passa para o estado ativo, quando todas as peças são recolocadas nos respectivos repositórios, ao lado da matriz com as posições de jogada. Quando ativado, o sistema aguarda a jogada do humano, que pode escolher uma peça de um dos bancos. Após realizar a jogada, o sistema passa a vez para o robô, que deliberará e aplicará sua jogada com uma peça do banco oposto. A mudança de uma peça que está sobre a matriz do jogo, ou a retirada de duas peças do banco, ou mesmo a movimentação de uma peça pelo adversário na vez do oponente, consistem, dentre outras, violações de jogada. As regras de validação da vitória ou empate seguem as definidas para o Jogo da Velha tradicional.

A Tabela 12 mostra as dimensões do código-fonte Java desenvolvido.

Tabela 12 – Dimensões do código-fonte desenvolvido em Java

Arquivos com código-fonte Java	Linhas
./br/usp/eesc/mecatronica/collaboration/events/BoardStatusEvent.java	28
./br/usp/eesc/mecatronica/collaboration/events/listeners/BoardStatusListener.java	16
./br/usp/eesc/mecatronica/collaboration/events/listeners/PositionListener.java	9
./br/usp/eesc/mecatronica/collaboration/events/listeners/PresenceListener.java	9
./br/usp/eesc/mecatronica/collaboration/events/listeners/UDPMessageListener.java	16
./br/usp/eesc/mecatronica/collaboration/events/PositionEvent.java	30
./br/usp/eesc/mecatronica/collaboration/events/PresenceEvent.java	21
./br/usp/eesc/mecatronica/collaboration/events/UDPMessageEvent.java	27
./br/usp/eesc/mecatronica/collaboration/forum/actors/Human.java	21
./br/usp/eesc/mecatronica/collaboration/forum/actors/ScaraRobot.java	154
./br/usp/eesc/mecatronica/collaboration/forum/Environment.java	11
./br/usp/eesc/mecatronica/collaboration/forum/ForumActivity.java	5
./br/usp/eesc/mecatronica/collaboration/forum/ForumActor.java	5
./br/usp/eesc/mecatronica/collaboration/forum/ForumObject.java	5
./br/usp/eesc/mecatronica/collaboration/forum/ForumSpace.java	5
./br/usp/eesc/mecatronica/collaboration/forum/ForumTime.java	5
./br/usp/eesc/mecatronica/collaboration/forum/objects/Gripper.java	63
./br/usp/eesc/mecatronica/collaboration/forum/objects/Link.java	26
./br/usp/eesc/mecatronica/collaboration/forum/objects/Rotatable.java	7
./br/usp/eesc/mecatronica/collaboration/forum/objects/RotationalLink.java	27
./br/usp/eesc/mecatronica/collaboration/forum/objects/TicTacToePiece.java	21
./br/usp/eesc/mecatronica/collaboration/forum/objects/Translatable.java	7
./br/usp/eesc/mecatronica/collaboration/forum/objects/TranslationalLink.java	28
./br/usp/eesc/mecatronica/collaboration/forum/objects/TranslationalRotationalLink.java	47
./br/usp/eesc/mecatronica/collaboration/forum/space/TicTacToeBoard.java	96
./br/usp/eesc/mecatronica/collaboration/game/EncodersMonitor.java	56
./br/usp/eesc/mecatronica/collaboration/game/TicTacToeGame.java	371
./br/usp/eesc/mecatronica/collaboration/io/Encoder.java	19
./br/usp/eesc/mecatronica/collaboration/io/MotorActuator.java	79
./br/usp/eesc/mecatronica/collaboration/io/PresenceSensor.java	48
./br/usp/eesc/mecatronica/collaboration/io/ServoMotor.java	61
./br/usp/eesc/mecatronica/collaboration/net/TCPMessageClient.java	59
./br/usp/eesc/mecatronica/collaboration/net/UDPMessageHandler.java	35
./br/usp/eesc/mecatronica/collaboration/net/UDPMessageMonitor.java	26
./br/usp/eesc/mecatronica/collaboration/net/UDPMessageReceiver.java	75
./br/usp/eesc/mecatronica/collaboration/resources/collaboration.properties	7
./br/usp/eesc/mecatronica/collaboration/rules/Block.java	65
./br/usp/eesc/mecatronica/collaboration/rules/BlockFork.java	195
./br/usp/eesc/mecatronica/collaboration/rules/Fork.java	67
./br/usp/eesc/mecatronica/collaboration/rules/PlayCenter.java	33
./br/usp/eesc/mecatronica/collaboration/rules/PlayEmptyCorner.java	36
./br/usp/eesc/mecatronica/collaboration/rules/PlayEmptySide.java	36
./br/usp/eesc/mecatronica/collaboration/rules/PlayOppositeCorner.java	39
./br/usp/eesc/mecatronica/collaboration/rules/RulesSelector.java	49
./br/usp/eesc/mecatronica/collaboration/rules/TicTacToeRule.java	70
./br/usp/eesc/mecatronica/collaboration/rules/Win.java	66
./br/usp/eesc/mecatronica/collaboration/ScaraCalibration.java	492
./br/usp/eesc/mecatronica/collaboration/TicTacToeBoardTest.java	754
./br/usp/eesc/mecatronica/collaboration/TicTacToeGameGUI.java	488
./br/usp/eesc/mecatronica/collaboration/UDPMessageTest.java	352
./br/usp/eesc/mecatronica/scara/Axis.java	124
./br/usp/eesc/mecatronica/scara/resources/Scara3D.properties	11
./br/usp/eesc/mecatronica/scara/Scara.java	133
./br/usp/eesc/mecatronica/scara/Scara3D.java	292
./br/usp/eesc/mecatronica/scara/ScaraRotationInterpolator.java	119
./br/usp/eesc/mecatronica/scara/ScaraTCPClient.java	60
./br/usp/eesc/mecatronica/scara/ScaraTCPServer.java	111
./br/usp/eesc/mecatronica/scara/ScaraTranslationInterpolator.java	76
./br/usp/eesc/mecatronica/scara/Tabuleiro.java	23
59 arquivos	5216

6.2.5. Testes e resultados

Por se tratar de um sistema complexo que envolve a integração com um robô industrial, surgiu a necessidade de se medir algumas características de seu desempenho. As principais características envolvidas referem-se à repetibilidade e à precisão do robô. Repetibilidade pode ser entendida como a medida que indica a capacidade do robô repetir o mesmo posicionamento, sempre que um comando for apresentado. Precisão refere-se à proximidade do TCP (*Tool Center Point*) do robô do ponto alvo que ele deve alcançar em determinado comando. Também foram realizados testes de operação de todo o sistema, tendo em vista a validação da aplicação das regras do jogo.

6.2.5.1. Testes de verificação: repetibilidade e precisão

Durante os procedimentos para o mapeamento das posições do tabuleiro, descritos na seção 6.2.4.2, notou-se certa instabilidade do robô. Tal instabilidade, decorrente de problemas de ajuste no PID do controlador (no *target*), afetou diretamente a repetibilidade do robô. Isso pode ser observado nos resultados dos mapeamentos dos pontos, depois de dois procedimentos de calibração distintos, nas Tabelas 13 e 14.

Tabela 13 – Mapeamento das posições antes do ajuste do PID

Local	Posição do link 1		Posição do link 2	
	Graus	Referência	Graus	Referência
1	10	8723	45	19731
2	13	11339	62	27185
3	15	13084	72	31570
4	27	23551	34	14908
5	28	24423	48	21047
6	18	15701	97	42531
7	30	26167	100	43847
8	43	37506	98	42970
9	28	24423	71	31131
10	36	31401	73	32008
11	50	43612	72	31570
12	41	35762	31	13593
13	48	41867	38	16662
14	61	53206	33	14470
15	57	49717	77	33762
16	67	58439	67	29377
17	78	68034	55	24116
18	62	54078	53	23239
19	75	65417	42	18416

Tabela 14 – Outro mapeamento das posições antes do ajuste do PID

Local	Posição do link 1		Posição do link 2	
	Graus	Referência	Graus	Referência
1	9	7851	51	22362
2	11	9595	67	29377
3	13	11339	78	34201
4	23	20062	40	17539
5	27	23551	52	22801
6	14	12212	102	44724
7	27	23551	106	46477
8	42	36634	104	45601
9	25	21806	75	32885
10	37	32273	79	34639
11	47	40995	78	34201
12	40	34889	37	16224
13	49	42739	42	18416
14	60	52334	39	17101
15	56	48845	82	35954
16	68	59312	73	32008
17	78	68034	59	25870
18	61	53206	59	25870
19	73	63673	48	21047

A partir da constatação dos problemas de instabilidade do robô, foram realizados ajustes dos parâmetros do PID do controlador, no *target*. O trabalho resultou em uma melhoria significativa da estabilidade do robô. Tal melhoria afetou a precisão e a repetibilidade do robô durante os novos procedimentos de mapeamento dos pontos que, ao final, foram fixados de acordo com os parâmetros apresentados na Tabela 15.

Tabela 15 – Mapeamento das posições depois do ajuste do PID

Local	Posição do link 1		Posição do link 2	
	Graus	Referência	Graus	Referência
1	14	12212	46	20170
2	15	13084	62	27185
3	18	15701	74	32447
4	29	25295	32	14031
5	29	25295	49	21485
6	18	15701	100	43847
7	30	26167	104	45601
8	44	38378	102	44724
9	29	25295	72	31570
10	38	33145	76	33324
11	50	43612	74	32447
12	46	40123	29	12716
13	53	46228	35	15347
14	63	54951	33	14470
15	59	51462	79	34639
16	69	60184	71	31131
17	80	69778	58	25431
18	64	55823	56	24554
19	75	65417	44	19293

Infelizmente, o mesmo resultado não pode ser observado no movimento linear do link 3. Durante os testes de reposicionamento das peças no tabuleiro, pode-se notar que o TCP distanciava-se gradativamente ao número de repetições de movimentos de translação realizados. Para tentar resolver o problema, foram alterados parâmetros que controlam a tarefa de tempo real no *target*.

Cada teste envolveu a configuração de três parâmetros distintos e a execução de dez movimentos consecutivos de translação entre os limites máximos do link. Os resultados para sete combinações diferentes são apresentados na Tabela 16 e representados no gráfico da Figura 38.

Pode-se notar que a alteração dos parâmetros fornece leituras de resultados instáveis e não determinísticos, e não acarreta em melhorias ao desempenho geral do sistema. Sendo assim, os resultados apresentaram-se inconclusivos para o ajuste de tais parâmetros o que aponta para a necessidade de avaliação mais detalhada do sistema de tempo real no *target*. Tal avaliação não está prevista no escopo deste trabalho.

Tabela 16 – Testes de repetibilidade do link 3

Tipo	Tick_period	Limiar	Delay	Diferença
1	5000000	30	500	2000
1	5000000	30	500	3500
2	5000000	50	0	2200
2	5000000	50	0	2200
2	5000000	50	0	2800
3	5000000	50	500	2000
3	5000000	50	500	2500
3	5000000	50	500	2200
3	5000000	50	500	3000
3	5000000	50	500	2200
3	5000000	50	500	2200
3	5000000	50	500	2200
3	5000000	50	500	2200
3	5000000	50	500	2200
4	15000000	50	0	1000
4	15000000	50	0	1000
4	15000000	50	0	2000
4	15000000	50	0	2000
4	15000000	50	0	2000
5	15000000	50	500	1500
5	15000000	50	500	1500
5	15000000	50	500	2000
6	15000000	100	500	3200
6	15000000	100	500	3900
6	15000000	100	500	3200
6	15000000	100	500	3800
7	25000000	50	500	2500
7	25000000	50	500	2800
7	25000000	50	500	2500

Figura 38 – Representação gráfica dos resultados dos testes de repetibilidade

6.2.5.2. Testes de validação: regras do jogo

A fim de se comprovar a funcionalidade do sistema e, principalmente, a deliberação das regras do jogo, foram realizados e registrados vários testes com o robô Scara. Três partidas do jogo são apresentadas nesta seção. Na Figura 39 são mostradas as cenas gravadas da primeira partida.

Figura 39 – Sequência de cenas gravadas em uma partida com o robô – 1

O humano iniciou a primeira partida e jogou uma peça 'O' no centro do tabuleiro. Nota-se que, como consequência de tal ação, o sistema deliberou a aplicação da regra “Jogar no canto livre”, e comandou o robô a pegar uma peça da posição 15 (banco de peças 'X') e jogar na posição 6, como mostra a Figura 40.

Figura 40 – Sequência correspondente de telas do sistema – 1

Nas jogadas que se revezaram a seguir, percebe-se que o humano tentou criar situações que permitissem sua vitória, ao colocar duas peças suas, primeiro na diagonal e, depois, na segunda linha. Em ambas as situações o sistema detectou a aplicabilidade da regra “Bloquear” e gerou as mensagens adequadas à respectiva operação do robô. Nessa partida que terminou empatada, pode-se notar a correta aplicação das regras “Jogar no canto livre” e “Bloquear”.

Em uma segunda partida, cujas cenas estão registradas na Figura 41, o humano realizou a primeira jogada, com uma peça do tipo ‘X’. Diferentemente da situação anterior, a peça foi colocada em um dos lados do tabuleiro. Em resposta a essa ação, o sistema conseguiu detectar a aplicabilidade da regra “Jogar no centro”, conforme mostra a Figura 42. Nota-se a mudança de comportamento do robô em relação à partida passada, pois a regra agora selecionada tem maior prioridade que a escolhida na partida anterior. Poder-se também notar a adaptação do jogo à mudança na escolha do tipo de peça, realizada pelo humano.

Em seguida, o humano colocou uma peça no canto inferior esquerdo, na tentativa de construir uma jogada que exigiria bloqueio duplo. Em resposta, o sistema deliberou a aplicação da regra “Intersectar” que, após comandar o reposicionamento da peça ao robô, permitiu o bloqueio da jogada em andamento. Nas jogadas que se sucederam, as tentativas de ataque do humano foram bloqueadas pelo sistema, o que levou a partida ao empate.

Figura 41 – Sequência de cenas gravadas em uma partida com o robô – 2

Apesar do novo empate ocorrido no final, a segunda partida mostrou situações que puderam comprovar a aplicabilidade de duas outras regras. Logo no início do jogo, a regra “Jogar no centro” foi corretamente selecionada e aplicada. E, na jogada seguinte, foi comprovada a correta utilização da jogada “Intersectar”.

Figura 42 – Sequência correspondente de telas do sistema – 2

Uma terceira partida é registrada nas Figuras 43 e 44. Nesta partida, o humano realiza a primeira jogada e coloca uma peça 'X' no centro do tabuleiro. Tal situação é semelhante à retratada na primeira partida e, como consequência, a mesma regra “Jogar no canto livre” é escolhida. Outra vez, o humano repete a jogada da primeira partida e, em seu segundo movimento, coloca uma peça no canto inferior esquerdo. Tal configuração remete o sistema à mesma situação anterior que o levou à aplicação da regra “Bloquear”. Em seguida, comandou o robô para jogar no canto superior direito.

Porém, desta vez o humano realizou na seqüência uma jogada errada, e colocou uma peça sua no canto inferior direito. A melhor jogada seria o bloqueio na primeira linha, dada a condição de vitória ao oponente na próxima jogada. O estado atual da partida, através do sensoramento do tabuleiro, permitiu o sistema detectar a aplicabilidade da regra “Vencer”, que foi comandada em seguida. O jogo terminou com vitória do robô. Portanto, neste teste foi demonstrada com sucesso a deliberação de outra regra, “Vencer”.

Figura 43 – Seqüência de cenas gravadas em uma partida com o robô – 3

Figura 44 – Sequência correspondente de telas do sistema – 3

Apesar dos testes não terem gerado situações para a aplicação de todas as regras, seus resultados permitiram demonstrar diversas funcionalidades do sistema. A primeira, evidentemente, refere-se ao dispositivo de seleção e de aplicação das regras. Outra constatação refere-se à correta seleção pelo sistema dos pontos de origem e de destino para o reposicionamento das peças, o que permitiu a representação fiel da jogada, tanto no ambiente real, como no virtual (simulador 3D).

Além disso, os testes foram também importantes para se demonstrar a integração de todo o sistema distribuído. O reposicionamento das peças magnéticas no tabuleiro foi detectado pelos sensores de presença e comunicado ao jogo via rede, como previsto, assim como a comunicação com o robô (real e virtual).

6.3. CONSIDERAÇÕES FINAIS

Neste capítulo buscou-se definir um modelo de regras adequado à representação da colaboração entre robôs e humanos. Pode-se constatar a viabilidade da aplicação do modelo M-Forum para descrever a política de controle e colaboração do ambiente.

Para testar a aplicabilidade do modelo na prática, um estudo de caso da colaboração robô-humano em um “jogo da velha” foi proposto, analisado e desenvolvido. As regras do jogo foram sumarizadas, analisadas e mapeadas em sintaxe L-Forum. Em seguida, foram apresentados os projetos dos componentes de *hardware* e de *software* definidos no desenvolvimento do sistema para o estudo de caso.

O modelo M-Forum também se mostrou adequado à representação por artefatos de *software* para o projeto do sistema do jogo. O projeto do sistema foi capaz de abstrair, além das representações dos elementos do ambiente, a integração com os agentes remotos (no *target*), e também todo o fluxo de controle, desde a entrada de dados dos sensores de posição e de presença até o comando da atuação dos motores. Foi então inserido no sistema o suporte às regras, com a implementação individual de cada regra do jogo e de um dispositivo deliberativo de seleção e aplicação.

Em seguida, descreveu-se a integração do sistema ao robô Scara e à sua simulação 3D, que representa o ambiente virtual descrito no capítulo 5. Um importante resultado foi demonstrado, quando se compararam as execuções de mesmos comandos em ambos os ambientes. Foi constatada grande fidelidade cinemática do modelo virtual, em relação ao robô Scara, o que reforça as conclusões prévias do capítulo 5.

Duas seções apresentaram testes distintos do sistema. Na primeira, relataram-se os testes de repetibilidade e de precisão do robô Scara (*target*). Os resultados dos testes apontaram para a necessidade de melhoria da implementação dos agentes locais que controlam cada motor do robô. No segundo conjunto de testes buscou-se validar a aplicação de regras pelo sistema. Foram apresentadas situações e os comportamentos deliberados pelo sistema, através das respectivas regras selecionadas. Os testes se mostraram importantes para a comprovação do uso das regras colaborativas, além do funcionamento geral do sistema.

Foram inicialmente vislumbrados testes com crianças, que motivaram a confecção da vestimenta de elefante para o robô. Porém, os problemas de instabilidade detectados no robô inviabilizaram essa interação no momento.

Pode-se portanto concluir que o uso de regras colaborativas permite a implementação no sistema de dispositivos que operam o comportamento deliberativo. A utilização do M-Forum ofereceu importantes subsídios ao projeto e à implementação do sistema, dada a prévia concepção dos elementos de um ambiente colaborativo e suas relações. A linguagem L-Forum apresentou recursos sintáticos suficientes para a expressão das regras utilizadas no estudo de caso.

Outra grande vantagem notada no uso das regras é a de isolar a lógica de cada situação do ambiente, o que favorece a manutenção do *software*. Além disso, operações de inclusão, remoção ativação ou desativação de regras podem ser facilmente realizadas. O dispositivo de seleção de regras pode também ser alterado para considerar outras condições da política do ambiente; esta modificação permite a definição futura de outras estratégias para a seleção de regras, em situações onde se deseja comportamentos mais adequados (como a adaptação do robô para errar jogadas, quando na interação com crianças).

7 CONCLUSÕES

Neste trabalho foi introduzido inicialmente o significado para colaboração em Robótica. Apesar da grande afinidade com a área de estudos de cooperação, que descreve a interação entre máquinas, o termo extrapola esse conceito. Assume-se por colaboração o relacionamento entre agentes robóticos e humanos e o ambiente que integram, para realização de uma meta comum. Essa meta e os elementos do ambiente colaborativo são representados na forma de regras simbólicas que consistem parte do modelo mental compartilhado entre os agentes.

A pesquisa na área da cooperação em Robótica não ofereceu subsídios para a sustentação completa do problema estudado. Isso se deve ao fato dos modelos tradicionalmente utilizados para a concepção do ambiente ser compartilhados apenas pelos robôs (ou seus módulos) envolvidos nas tarefas cooperativas. Sendo assim, fez-se necessária a definição de um modelo mental que permitisse o compartilhamento entre robôs e seres humanos.

Para tal propósito, foi apresentada uma visão geral sobre o estudo da mente humana a partir de concepções oriundas da Filosofia e da Psicologia, bem como as suas diferentes estratégias de abstração em ambientes computacionais. As duas abordagens aplicadas à abstração da interpretação, conexionista e simbólica, foram apresentadas e comparadas. Concluiu-se desse estudo que a abordagem simbólica é a mais adequada para o mapeamento das regras em um formato compartilhado entre o agente humano e o computador (que comanda o agente robótico).

Também foram pesquisadas as estratégias de controle de robôs e pode-se notar que o modelo adaptativo em camadas, proposto por Brooks (1986), é mais adequado à concepção de arquiteturas distribuídas, ideais para o controle de múltiplos agentes. Porém, esse modelo é puramente reativo, o que não atende à demanda pelo comportamento deliberativo desejada para a solução do problema proposto. Além disso, o modelo desejado para este trabalho deve também permitir a integração do modelo de regras cooperativas e colaborativas. Sendo assim, um novo modelo de controle distribuído, que permite a deliberação baseada em tais regras, é proposto e apresentado neste trabalho, no capítulo 5.

Em seguida, foram pesquisados os recursos que a área da Computação dispõe para permitir a abstração de ambientes colaborativos. Em linhas gerais, as regras definem políticas que regem o ambiente colaborativo, e descrevem elementos nele presentes. O principal desafio enfrentado foi a escolha de um modelo que permitisse o mapeamento das regras para ambientes colaborativos que integram robôs e humanos. O estudo indicou que o modelo M-Forum e sua linguagem formal L-Forum ofereciam maior suporte às exigências do problema em questão e, portanto, foram adotados neste trabalho.

No capítulo 5 deste documento foi proposto e apresentado um novo modelo de arquitetura para o controle distribuído de robôs, que pode ser aplicado a partes distintas de um manipulador, e que permite a inclusão de comportamentos cooperativos e colaborativos. Os resultados apresentados permitiram constatar que é possível a integração entre os agentes globais e locais previstos pelo modelo através de redes de comunicação de dados. Tal integração pode ser demonstrada por uma interface gráfica 3D, desenvolvida para representar o ambiente virtual do modelo proposto.

No capítulo 6 foi abordada e demonstrada a modelagem de regras sintáticas de colaboração entre robôs e humanos. Os elementos previstos nas regras foram abstraídos no modelo do sistema desenvolvido para um estudo de caso que envolveu a colaboração entre um robô manipulador e seres humanos em um “jogo da velha”. As regras do jogo foram delineadas a partir do modelo algorítmico apresentado por Crowley e Siegler (1993). Neste ponto registra-se outra inovação deste trabalho, ao se traduzir esse modelo para o formalismo do M-Forum.

Além disso, com a integração do sistema ao robô Scara e ao simulador Scara3D foi constatada a grande fidelidade do modelo cinemático do simulador ao ambiente real do robô. Tal resultado reforça a importância do ambiente virtual da arquitetura proposta no capítulo 5. Testes de operação do robô foram realizados e pode-se constatar da análise dos seus resultados que o sistema de acionamento apresentou problemas de estabilidade e de repetibilidade, que apontam para a necessidade de sua melhoria. Também foram realizados testes com o sistema completo, que permitiram comprovar e validar a deliberação dos comportamentos previstos, mediante processos de seleção e aplicação das regras. Estes testes também permitiram constatar a integração das partes do sistema distribuído, de acordo com os modelos inicialmente vislumbrados.

Ao analisar o estudo de caso descrito no capítulo 6, pode-se observar a colaboração entre os atores, robô e humano. Tal colaboração se dá pela complementação de papéis, descritos como adversários em um jogo. Ambos detêm as mesmas noções sobre os elementos envolvidos nesse ambiente, como adversários (atores), peças (objetos), tabuleiro (espaço) e partida (tempo). Além disso, compartilham relações de causa-efeito entre os elementos e deliberam ações (atividades), de acordo com as regras preestabelecidas.

As regras modeladas para o jogo descrevem um conjunto limitado de situações de interação entre os adversários, mesmo porque é impossível prever todo o comportamento humano. Para a solução desse tipo de problema, outras condições de contorno devem ser analisadas, o que fornece subsídios para a adição de novas regras ao sistema. Nesse sentido, até primitivas de segurança para a interação com humanos podem ser descritas no modelo de regras. Tal incremento de novas situações ao repertório de regras é também útil em situações de conflito. Pois, ao se detectar que a aplicação de uma regra invalida o estado de outra, com maior relevância (como em uma violação de segurança), pode-se decidir pela não aplicação daquela regra.

Conclui-se, enfim, que o modelo de arquitetura proposto pode ser implementado de forma a permitir a integração de regras para a deliberação de comportamentos colaborativos. A utilização de um modelo já consolidado como o M-Forum para a concepção das regras colaborativas forneceu importantes contribuições para a modelagem e o desenvolvimento das regras. A aplicação das regras é especialmente importante em sistemas que envolvem modelos previamente concebidos e compartilhados por seres humanos. O uso de regras colaborativas permite abstração extra ao sistema, e o torna mais flexível e adaptável em tais ambientes.

7.1. SUGESTÕES PARA TRABALHOS FUTUROS

Uma das oportunidades vislumbradas para futuros trabalhos consiste em modificar a estratégia de seleção de regras do sistema para permitir que o robô as aplique em ordem diferente da prioridade definida. Isso pode permitir a implementação de diferentes níveis de dificuldade, que pode ser importante para o robô interagir pessoas com diferentes níveis de habilidade (de crianças a adultos).

Em outra melhoria, o sistema pode ser alterado para armazenar os dados sobre seleção de jogadas realizadas pelo humano. Isso pode permitir o aprendizado de estratégias de seleção mais adaptadas aos padrões de jogada realizados por seres humanos. Além disso, se o sistema aprender e armazenar modelos diferentes de jogadas poderá apoiar o treinamento de outras pessoas futuramente (para ensinar o jogo, ou outras atividades colaborativas, p.e).

Estratégias de aprendizado podem também ser aplicadas para a inferência de novas regras. A integração com bases de conhecimento de senso comum (OpenMind, p.e) permitiria a definição de novos comportamentos adequados ao ambiente colaborativo. Tais regras podem ser analisadas e carregadas pelo sistema durante seu funcionamento, o que consiste uma oportunidade de integração com a área da Computação Autônoma. A própria detecção do humano, hoje realizada no sistema pela leitura de um pequeno botão, pode ser melhorada. Técnicas de identificação de imagens da área de visão computacional podem ser utilizadas para a detecção de seres humanos e suas características (identificação de gestos, sexo e faixa etária, p.e). Outros sensores, como barreiras óticas poderiam também permitir maior segurança na interação robô-humano.

Os problemas notados durante os testes e ajustes do robô, para melhoria de parâmetros de repetibilidade e de precisão, inspiraram outros possíveis trabalhos futuros. Uma das possibilidades refere-se à melhoria nos componentes de controle do robô Scara, que poderia ser submetido a outro processo de *retrofitting*. Outra possibilidade envolve a integração no sistema de um robô industrial mais novo, dado que atualmente os fabricantes como a Motoman oferecem soluções (MotoCom SDK) para comunicação e operação via rede (Ethernet com TCP/IP). E, ao se adotar uma solução que permita melhor desempenho do robô, outros testes podem ser realizados, inclusive os que envolvem tempos de latência na tele-operação do *target*, em situações distintas quanto à distância ao sistema.

Outra consideração que deve ser registrada é que não foi formalizado e aplicado um plano de testes ao sistema, o que permitiria verificar e validar o seu funcionamento adequadamente. Sugere-se que em futuros testes de validação sejam envolvidas crianças, desde que o robô permita operação segura, e assim estender o estudo deste trabalho a outras áreas, como a Educação e a Pedagogia.

Durante a elaboração deste trabalho não foram identificados outros projetos que tenham aplicado a linguagem L-Forum à implementação de sistemas de *software* deliberativos. Sendo assim, sugere-se para os futuros trabalhos nessa linha que comparem seus resultados com os aqui demonstrados.

Por fim, é sugerida também a aplicação do conceito de colaboração aqui introduzido, bem como a sua abstração computacional proposta, em problemas que não consistem jogos. Os problemas que integram esse perfil são aqueles que demandam a modelagem de comportamentos para os quais existem regras e normas já instituídas. Podem ser citados, como exemplos, os modelos de interação social (robôs humanóides no convívio com humanos, p.e), e os modelos de produção empresariais (integração de robôs e humanos em linhas de produção, p.e).

REFERÊNCIAS

- ADLER, D. Reverse Engineering the Brain. **IEEE Spectrum**, New York, v. 45, n. 6, p. 43-45, Jun. 2008.
- AROCA, R.; TAVARES, D. M.; CAURIN, G. A. P. Scara robot controller using real time linux. In: International Conference on Advanced Intelligent Mechatronics (IEEE/ASME), 2007, Zürich, Switzerland. **Proceedings of the 2007 IEEE/ASME International Conference on Advanced Intelligent Mechatronics**, Zurich: IEEE, 2007. p. 1-6.
- AROCA, R. V. **Análise de sistemas operacionais de tempo real para aplicações de robótica e automação**. 2008. Dissertação (Mestrado) – Departamento de Engenharia Mecânica, Universidade de São Paulo, São Carlos, 2008.
- ASIMOV, I. **Eu, robô**. 10. ed. Rio de Janeiro: Exped, 1978. 296 p.
- BACKUS, J. W. et al. Revised Report on the Algorithmic Language ALGOL 60. **Communications of the ACM**, New York, v. 6, p. 1-17, Jan. 1963.
- BEAR, M. F.; CONNORS, B.; PARADISO, M. **Neuroscience: Exploring the Brain**. 2nd ed. New York: Lippincott Williams & Wilkins, 2002. p. 240-269.
- BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. The Semantic Web. **Scientific American**, New York, Mai. 2001.
- BIMBER, O. Total Recall. **IEEE Computer**, New York, v.41, n. 10, p. 32-33, Oct. 2008.
- BREAZEL, C.; HOFFMAN, G.; LOCKERD, A. Teaching and Working with Robots as a Collaboration. In: International Conference on Autonomous Agents (AAMAS), 2004, New York, EUA. **Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems – v. 3**, New York: ACM, 2004. p. 1028-1035.
- BROOKS, R. A. A robust layered control system for a mobile robot. **IEEE Journal of Robotics and Automation**, New York, v. 2, p. 14-23, Mar. 1986.
- BROOKS, R. A. Intelligence Without Reason. In: International Joint Conference on Artificial Intelligence (IJCAI), 12., 1991, Sydney, Australia. **Proceedings of the Twelveth International Joint Conference on Artificial Intelligence (IJCAI-91)**. San Francisco: Morgan Kaufmann, 1991. p. 569-595.
- BROOKS, R. A. I, Rodney Brooks, am a Robot. **IEEE Spectrum**, New York, v. 45, n. 6, p. 63-67, Jun. 2008.
- CAMOLESI Jr, L.; MARTINS, L. E. G. A Model for Interaction Rules to Define Governance Policies in Collaborative Environments. **Computer Supported Cooperative Work in Design II**, v. 3865/2006, p. 11-20, 2006. Revised Selected Paper from 9th International Conference, CSCWD 2005.

CAO, Y. U.; FUKUNAGA, A. S.; KAHNG, A. B. Cooperative Mobile Robotics: Antecedents and Directions. **Autonomous Robots**, Dordrecht, v. 4, p. 7-27, Mar. 1997.

CAURIN, G. A. P.; ALBUQUERQUE, A. R. L.; MIRANDOLA, A. L. A. Manipulation strategy for an anthropomorphic robotic hand. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2004, Sendai, Japan. **Proceedings of IROS 2004 – IEEE/RSJ International Conference on Intelligent Robots and Systems**. Sendai: IEEE, 2004. p. 1656-1661.

CAZANGI, R. R.; ZUBEN, F. J. V.; FIGUEIREDO, M. F. Autonomous Navigation System Applied to Collective Robotics with Ant-Inspired Communication. In: Genetic And Evolutionary Computation Conference, 2005, Washington DC, USA. **Proceedings of the 2005 conference on Genetic and evolutionary computation**. Washington: Association for Computing Machinery, 2005. p. 121-128.

CHAIMOWICZ, L.; SUGAR, T.; KUMAR, V.; CAMPOS, M. F. M. An Architecture for Tightly Coupled Multi-Robot Cooperation. In: IEEE International Conference on Robotics and Automation (ICRA), 2001, Seoul, Korea. **Proceedings 2001 ICRA – IEEE International Conference on Robotics and Automation**. Seoul: IEEE, 2001. p. 2992-2997.

CHAKRABORTY, N.; SYCARA, K. Reconfiguration Algorithms for Mobile Robotic Networks. In: IEEE International Conference on Robotics and Automation (ICRA), 2010, Anchorage, Alaska. **Proceedings 2010 ICRA – IEEE International Conference on Robotics and Automation**. Anchorage: IEEE, 2010. p. 5484-5489.

CHOMSKY, N. **Linguagem e Mente**. 3. ed. São Paulo: Editora UNESP, 2009. 342 p.

COMPTON, K.; HAUCK, S. Reconfigurable Computing: A Survey of Systems and Software. **ACM Computing Surveys**, New York, v. 34, n. 2, p. 171-210, Jun. 2002.

COREN, S.; WARD, L. M.; ENNS, J. T. **Sensation and Perception**. 5th ed. New York: Harcourt Brace College Publishers, 1999. p. 284-321.

CORRELL, N.; RUTISHAUSER, S.; MARTINOLI, A. Comparing Coordination Schemes for Miniature Robotic Swarms: A Case Study in Boundary Coverage of Regular Structures. International Symposium on Experimental Robotics (ISER '06), 10., 2006, Rio de Janeiro, Brasil. **Electronic Proceedings of ISER 2006 10th International Symposium on Experimental Robotics**. Rio de Janeiro: ISER, 2006. CD-ROM.

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. **Sistemas Distribuídos: Conceitos e Projeto**. 4. ed. Porto Alegre: Bookman, 2007. 792 p.

CRAIG, J. J. **Introduction to Robotics: Mechanics and Control**. 3. ed. New Jersey: Prentice Hall, 2004. 408 p.

CROWLEY, K.; SIEGLER, R. S. Flexible strategy use in young children's tic-tac-toe. **Cognitive Science**, v. 17, p. 531-561, Out. 1993.

DEHON, A.; WAWRZYNEK, J. Reconfigurable Computing: What, Why, and Implications for Design Automation. Design Automation Conference, 36., 1999, New Orleans, USA. **Proceedings of the 36th ACM/IEEE Conference on Design Automation (DAC'99)**. New Orleans: ACM/IEEE, 1999. p. 610-615.

DIVERIO, T. A.; MENEZES, P. B. **Teoria da computação: máquinas universais e computabilidade**. 2. ed. Porto Alegre: Sagra Luzzatto, 2000. 224 p.

DRURY, J. L.; SCHOLTZ, J.; YANCO, H. A. Awareness in Human-Robot Interactions. In: IEEE International Conference on Systems, Man and Cybernetics, 2003, Washington, D.C., USA. **Proceedings of the IEEE Conference on Systems, Man and Cybernetics**. Washington: IEEE, 2003. p. 912-918.

DUDEK, G.; JENKIN, M.; MILIOS, E. A Taxonomy of Multirobot Systems. In: BALCH, T.; PARKER, L. E. **Robot Teams: From Diversity to Polymorphism**. Massachusetts: A K Peters, 2002. p. 3-22.

EDWARDS, W. K. Policies and Roles in Collaborative Applications. In: The 1996 ACM Conference on Computer Supported Cooperative Work (CSCW), 1996, Cambridge, EUA. **Proceedings of ACM Conference on Computer-Supported Cooperative Work (CSCW'96)**. New York: ACM, 1996. p. 11-20.

FARIA, G.; PRESTES, E.; IDIART, M. A. P.; ROMERO, R. A. F. Multi Robot System based on Boundary Value Problems. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2006, Beijing, China. **Proceedings of IROS 2006 – IEEE/RSJ International Conference on Intelligent Robots and Systems**. Beijing: IEEE, 2006. p. 2065-2069.

FODOR, J. A. The Mind-Body Problem. **Scientific American**, New York, v. 244, p. 114-123, Jan. 1981.

FISCHER, K.; MÜLLER, J. P.; PISCHEL, M. **Unifying Control in a Layered Agent Architecture**. Saarbrücken: DFKI – GmbH, 1996. 29 p. (Technical Report, TM-94-05)

GARCIA, E.; JIMENEZ, M. A.; DE SANTOS, P. G.; ARMANDA, M. The Evolution of Robotics Research: From Industrial Robotics to Field and Service Robotics. **Robotics & Automation Magazine**, New York, v. 14, p. 90-103, Mar. 2007.

GEBARA, M. **Regulação Epigenética e Psicofenótipo**. Publicado em: 15 ago. 2009. Disponível em: <http://marcosgebara.blogspot.com/2009/08/regulacao-epigenetica-e-psicofenotipo.html>. Acesso em: 23 ago. 2010.

GERKEY, B. P.; MATARIĆ, M. J. A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems. **The International Journal of Robotics Research**, Thousand Oaks, USA, v. 23, p. 939-954, Sep. 2004.

GIL, S.; SCHWAGER, M.; JULIAN, B. J.; RUS, D. Optimizing Communication in Air-Ground Robot Networks using Decentralized Control. In: IEEE International Conference on Robotics and Automation (ICRA), 2010, Anchorage, Alaska. **Proceedings 2010 ICRA – IEEE International Conference on Robotics and Automation**. Anchorage: IEEE, 2010. p. 1964-1971.

GOETZ, A. T.; SHACKELFORD, T. K. **Introduction to Evolutionary Theory and Its Modern Application to Human Behavior and Cognition**. In: PLATEK, S. M.; KEENAN, J. P.; SHACKELFORD, T. K. (Eds.). *Evolutionary cognitive neuroscience*. Cambridge, MA: MIT Press, 2007. chap. 1, p. 5-19.

GÓMEZ-PÉREZ, A.; FERNÁNDEZ-LÓPEZ, M.; CORCHO, O. **Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web**. 1. ed. London: Springer-Verlag, 2004. p. 6-9.

GROTEN, R.; FETH, D.; PEER, A.; BUSS, M. Shared Decision Making in a Collaborative Task with Reciprocal Haptic Feedback – an Efficiency-Analysis. In: IEEE International Conference on Robotics and Automation (ICRA), 2010, Anchorage, Alaska. **Proceedings 2010 ICRA – IEEE International Conference on Robotics and Automation**. Anchorage: IEEE, 2010. p. 1834-1839.

GRUBER, T. R. A Translation Approach to Portable Ontology Specifications. **Knowledge Acquisition**, London, v. 5, p. 199-220, Jun. 1993.

GRUDIN, J. CSCW: Time Passed, Tempest, and Time Past. **Interactions Magazine (ACM)**, New York, v. 17, p. 38-40, Jul/Ago. 2010.

GUTWIN, C.; GREENBERG, S. **The importance of awareness for team cognition in distributed collaboration**. Report 2001-696-19, University of Calgary, 2001. Disponível em: <http://www.cpsc.ucalgary.ca/grouplab/papers/>. Acesso em: 23 ago. 2010.

HALLOY, J. et al. Social Integration of Robots into Groups of Cockroaches to Control Self-Organized Choices. **AAAS Science**, New York, v. 318, n. 5853, p. 1155-1158, Nov. 2007.

HAYKIN, S. **Neural Networks: A Comprehensive Foundation**. 2. ed. New Jersey: Prentice Hall, 1998. 842 p.

HERRERA, J. A. F.; MARTINS Jr, J.; NOGUEIRA Neto, M. S.; MIHO, M. A.; CAURIN, G. A. P. A Soft-PLC Model Approach for Generic Device Controllers. In: IASTED International Conference on Modelling and Simulation, 14., 2003, Palm Springs, USA. **Proceedings of the 14th IASTED International Conference on Modelling and Simulation**. Palm Springs: Acta Press, 2003.

HOLLAND, O. From the Imitation of Life to Machine Consciousness. In: GOMI, T. **Evolutionary Robotics. From Intelligent Robotics to Artificial Life: International Symposium, ER 2001, Tokyo, Japan, October 18-19, 2001. Proceedings (Lecture Notes in Computer Science)**. Berlin: Springer, 2001. p. 1-37.

HOUAISS, A. **Dicionário Eletrônico Houaiss da Língua Portuguesa**. 1. ed. Rio de Janeiro: Objetiva, 2001.

HUHNS, M. N. Software Agents: The Future of Web Services. In: **Lecture Notes in Computer Science**, Vol. 2592, CARBONELL, J. G.; SIEKMANN, J.; KOWALCZYK, R.; MÜLLER, J. P.; TIANFIELD, H.; UNLAND, R. (Eds.), New York: Springer, 2003. p. 1-18.

HUHNS, M. N.; SINGH, M. P. Ontologies for Agents. **IEEE Internet Computing**, New York, v. 1, p. 81-83, Nov. 1997.

ICARCV – International Conference on Control, Automation, Robotics and Vision, 8., 2004, Kunming, China. **2004 8th International Conference on Control, Automation, Robotics and Vision (ICARCV) – vols. 1, 2, 3**. Kunming: IEEE, 2004. 2336 p.

ISER, R.; MARTENS, A.; WAHL, F. M. Localization of Mobile Robots Using Incremental Local Maps. In: IEEE International Conference on Robotics and Automation (ICRA), 2010, Anchorage, Alaska. **Proceedings 2010 ICRA – IEEE International Conference on Robotics and Automation**. Anchorage: IEEE, 2010. p. 4873-4880.

ITTELSON, W. H. Environment perception and contemporary perceptual theory. In: ITTELSON, W. H. (Ed.), **Environment and cognition**. New York: Seminar Press, 1973. p. 141-154.

JOBLING, C. P.; GRANT, P. W.; BARKER, H. A.; TOWNSEND, P. Object-oriented programming in control system design: a survey. **Automatica (Journal of IFAC)**, Tarrytown, NY, USA, v. 30, p. 1221-1261, Aug. 1994.

KEMP, C. C.; EDSINGER, A.; TORRES-JARA, E. Challenges for robot manipulation in human environments: Grand Challenges of Robotics. **Robotics & Automation Magazine**, New York, v. 14, p. 20-29, Mar. 2007.

KITANO, H.; ASADA, M.; KUNIYOSHI, Y.; NODA, Y. I.; OSAWA, E. RoboCup: The Robot World Cup Initiative. In: International Conference on Autonomous Agents, 1., 1997, Marina del Rey, USA. **Proceedings of the first international conference on Autonomous agents**. Washington: Association for Computing Machinery, 1997. p. 340-347.

KOCH, C.; TONONI, G. Can Machines be Conscious? **IEEE Spectrum**, New York, v. 45, n. 6, p. 47-51, Jun. 2008.

LAU, H. Y. K.; NG, A. K. S. Immunology-based Motion Control for Modular Hyper-redundant Manipulators. In: IFAC World Congress, 16., 2005, Prague. **Proceedings of the 16th IFAC World Congress**. New York: Elsevier, 2005.

LEBEDEV, M. A.; NICOLELIS, M. A. L. Brain-machine interfaces: past, present and future. **Trends in Neurosciences**, New York, v. 29, n. 9, p. 536-546, sep. 2006.

MARTINS, L. E. G.; CAMOLESI Jr, L. FORUM: Modelo e Linguagem para Especificação de Regras em Ambientes Colaborativos. **Revista de Informática Teórica e Aplicada**, Porto Alegre, v. 15, p. 9-26, nº 2, 2008.

MARTINS Jr, J.; CAMOLESI Jr, L.; CAURIN, G. A. P. Scara3D: 3-Dimensional HRI integrated to a distributed control architecture for remote and cooperative actuation. In: Applied Computing 2008 – Annual ACM Symposium on Applied Computing – Session: Robotics: hardware, software, and embedded systems, 23., 2008, Fortaleza, Brasil. **Proceedings of the 23rd Annual ACM Symposium on Applied Computing (SAC 2008)**, New York: ACM, 2008. p. 1597-1601.

MARTINS Jr, J.; CAMOLESI Jr, L.; CAURIN, G. A. P. **Collaborative rules operating manipulators**. In: HALL, E. (Ed.). Advances in Robot Manipulators. Vienna: In-Tech, 2010. cap. 7, p. 133-154.

MCFARLAND, D. J.; WOLPAW, J. R. Brain-Computer Interface Operation of Robotic and Prosthetic Devices. **IEEE Computer**, New York, v.41, n. 10, p. 52-56, Oct. 2008.

MINSKY, M. Logical vs. analogical or symbolic vs. connectionist or neat vs. scruffy. In: WINSTON, P. H.; SHELLARD, S. A. **Artificial Intelligence at MIT, Vol. 1**. Cambridge: MIT Press, 1990. p. 218-243.

MINSKY, M. Consciousness. In: MINSKY, M. **The Emotion Machine** (book draft). 2006. Disponível em: <http://web.media.mit.edu/~minsky/>. Último acesso em: 08 de abril de 2010.

MONDADA, F.; GAMBARDELLA, L. M.; FLOREANO, D.; NOLFI, S.; DENEUBOURG, J.; DORIGO, M. The Cooperation of Swarm-Bots: Physical Interactions in Collective Robotics. **Robotics & Automation Magazine**, New York, v. 12, p. 21-28, Jun. 2005.

MURATA, S.; YOSHIDA, E.; KAMIMURA, A.; KUOKAWA, H.; TOMITA, K.; KOKAJI, S. M-TRAN: Self-Reconfigurable Modular Robotic System. **IEEE/ASME Transactions on Mechatronics**, New York, v. 7, p. 431-441, Dec. 2002.

NAJMAEI, N.; KERMANI, M. R.; Prediction-Based Reactive Control Strategy for Human-Robot Interactions. In: IEEE International Conference on Robotics and Automation (ICRA), 2010, Anchorage, Alaska. **Proceedings 2010 ICRA – IEEE International Conference on Robotics and Automation**. Anchorage: IEEE, 2010. p. 3434-3439.

NICOLELIS, M. A. L. Action from thoughts. **Nature**, New York, v. 409, p. 403-407, Jan. 2001.

NWANA, H. S. Software Agents: An Overview. **Knowledge Engineering Review**, New York, v. 11, p. 205-244, Oct.-Nov. 1996.

PARK, C. H.; HOWARD, A. M. Understanding a Child's play for Robot Interaction by Sequencing Play Primitives using Hidden Markov Models. In: IEEE International Conference on Robotics and Automation (ICRA), 2010, Anchorage, Alaska. **Proceedings 2010 ICRA – IEEE International Conference on Robotics and Automation**. Anchorage: IEEE, 2010. p. 170-177.

PARK, C. H.; YOO, J. W.; HOWARD, A. M. Transfer of Skills between Human Operators through Haptic Training with Robot Coordination. In: IEEE International Conference on Robotics and Automation (ICRA), 2010, Anchorage, Alaska. **Proceedings 2010 ICRA – IEEE International Conference on Robotics and Automation**. Anchorage: IEEE, 2010. p. 229-235.

PARKER, L. E. ALLIANCE: An Architecture for Fault Tolerant Multirobot Cooperation. **IEEE Transactions on Robotics and Automation**, New York, v. 14, p. 220-240, Apr. 1998.

PARKER, L. E. Current research in multirobot systems. **Artificial Life and Robotics**, Tokyo, v. 7, n. 1-2, p. 1-5, Mar. 2003.

PEI, Y.; MUTKA, M. W.; XI, N. Coordinated Multi-Robot Real-Time Exploration With Connectivity and Bandwidth Awareness. In: IEEE International Conference on Robotics and Automation (ICRA), 2010, Anchorage, Alaska. **Proceedings 2010 ICRA – IEEE International Conference on Robotics and Automation**. Anchorage: IEEE, 2010. p. 5460-5465.

PINKER, S. **Como a Mente Funciona**. 2. ed. São Paulo: Companhia das Letras, 1998. 666 p.

PINKER, S. **O Instinto da Linguagem: como a mente cria a linguagem**. São Paulo: Martins Fontes, 2002. 627 p.

PROROK, A.; CIANCI, C. M.; MARTINOLI, A. Towards Optimally Efficient Field Estimation with Threshold-Based Pruning in Real Robotic Sensor Networks. In: IEEE International Conference on Robotics and Automation (ICRA), 2010, Anchorage, Alaska. **Proceedings 2010 ICRA – IEEE International Conference on Robotics and Automation**. Anchorage: IEEE, 2010. p. 5453-5459.

SHAPIRO, L. G.; STOCKMAN, G. C. **Computer Vision**. New Jersey: Prentice Hall, 2001. p. 13-32.

SLONNEGER, K.; KURTZ, B. L. **Formal Syntax and Semantics of Programming Languages: a laboratory based approach**. New York: Addison-Wesley Publishing Company, 1995. p. 1-30.

SOMMERVILLE, I. **Engenharia de Software**. 6. ed. São Paulo: Addison Wesley, 2003. 592 p.

STAMPER, R. et al. Understanding the Roles of Signs and Norms in Organisations - A semiotic approach to information systems design. **Journal of Behaviour & Information Technology**, London, v. 19, n. 1, p. 15-27, 2000.

TANENBAUM, A. S. **Redes de Computadores**. 4. ed. Rio de Janeiro: Elsevier, 2003. 945 p.

TAVARES, D. M.; AROCA, R.; CAURIN, G. A. P. Upgrade of a scara robot using orocos. In: IASTED International Conference on Robotics and Applications, 13., 2007, Würzburg, Germany. **Proceedings of the 13th IASTED International Conference on Robotics and Applications (RA 2007)**. Würzburg: Acta Press, 2007. p. 252-257.

TEIXEIRA, J. F. **Mentes e Máquinas: Uma introdução à ciência cognitiva**. Porto Alegre: Artmed, 1997. 182 p.

TONTI, G.; BRADSHAW, J. M.; JEFFERS, R.; MONTANARI, R.; SURI, N.; USZOK, A. Semantic Web Languages for Policy Representation and Reasoning: A Comparison of KAoS, Rei, and Ponder. In: **Lecture Notes in Computer Science**, Vol. 2870, FENSEL, D.; SYCARA, K.; MYLOPOULOS, J. (Eds.), Heidelberg: Springer Berlin, 2003. p. 419-437.

VINGE, V. Signs of the Singularity. **IEEE Spectrum**, New York, v. 45, n. 6, p. 69-74, Jun. 2008.

VIVALDINI, K. C. T.; GALDAMES, J. P. M.; PASQUAL, T. B.; BECKER, M.; CAURIN, G. A. P. **Intelligent Warehouses: Focus on the Automatic Routing and Path Planning of Robotic Forklifts Able to Work Autonomously**. In: DUTRA, M. S.; PÉREZ, O. L. (Eds.). *Mechatronics Systems: Intelligent Transportation Vehicles*. 1 ed. Brussels: Bentham Science Publishers Ltd., 2010, v. 7, p. 110-139.

VIVALDINI, K. C. T.; GALDAMES, J. P. M.; PASQUAL, T. B.; BECKER, M.; CAURIN, G. A. P. Robotic Forklifts for Intelligent Warehouses: Routing, Path Planning, and Auto-localization. In: IEEE International Conference on Industrial Technology (ICIT), 2010, Viña Del Mar, Chile. **Proceedings 2010 ICIT – International Conference on Industrial Technology**. Viña Del Mar: IEEE, 2010. v. 1. p. 1463-1468.

WANG, Z.; NAKANO, E.; TAKAHASHI, T. Solving function distribution and behavior design problem for cooperative object handling by multiple mobile robots. **IEEE Transactions on Systems, Man and Cybernetics, Part A**, New York, v. 33, p. 537-549, Sep. 2003.

YANCO, H.A.; DRURY, J. Classifying Human-Robot Interaction - An Updated Taxonomy. In: IEEE International Conference on Systems, Man and Cybernetics, 2004, The Hague, Netherlands. **Proceedings of the IEEE Conference on Systems, Man and Cybernetics**. The Hague: IEEE, 2004. p. 2841-2846.

YIM, M.; DUFF, D. G.; ROUFAS, K. D. PolyBot: a modular reconfigurable robot. In: IEEE International Conference on Robotics and Automation (ICRA), 2000, San Francisco, USA. **Proceedings ICRA'00 – IEEE International Conference on Robotics and Automation**. San Francisco: IEEE, 2000. p. 514-520.

YIM, M.; ZHANG, Y.; DUFF, D. G. Modular Robots. **IEEE Spectrum**, New York, v. 39, p. 30-34, Feb. 2002.

YIM, M. et al. Modular Self-Reconfigurable Robot Systems – Grand Challenges of Robotics. **Robotics & Automation Magazine**, New York, v. 14, p. 43-52, Mar. 2007.