

**UNIVERSIDADE FEDERAL DE SANTA CATARINA
GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA**

JAN ANTONIO PEREIRA

**ESTUDO SISTEMÁTICO DE AUXÍLIO À IMPLEMENTAÇÃO DE UM TIME DE
FUTEBOL DE ROBÔS SIMULADOS NO SIMULADOR ROBOCUP**

**Prof. Mauro Roisenberg
Orientador**

Florianópolis, 2004

JAN ANTONIO PEREIRA

**ESTUDO SISTEMÁTICO DE AUXÍLIO À IMPLEMENTAÇÃO DE UM TIME DE
FUTEBOL DE ROBÔS SIMULADOS NO SIMULADOR ROBOCUP**

Monografia apresentada para obtenção do grau de
Bacharelado em Ciências da Computação na
Universidade Federal de Santa Catarina.

Professor Coordenador : Renato Cislighi

Professor Orientador : Mauro Roisenberg

Membros da Banca : Antônio Carlos Mariani

Jorge Muniz Barreto

Florianópolis, 2004

AGRADECIMENTOS

Agradeço primeiramente a minha família , em especial a minha mãe e a minha namorada pelo apoio que me reservaram nos momentos mais difíceis durante a realização deste trabalho.

Ao meu orientador Prof. Mauro Roisenberg, pela indicação do tema proposto e colaboração no decorrer do trabalho.

Aos colegas de turma , pela amizade e companheirismo demonstrados durante todo o curso.

E a todos que direta ou indiretamente contribuíram para esse trabalho.

SUMÁRIO

Lista de Tabelas	VII
Lista de Ilustrações	VIII
Resumo	IX
Abstract	X
1 INTRODUÇÃO	11
1.1 JUSTIFICATIVA	11
1.2 OBJETIVOS.....	12
1.2.1 Objetivo Geral	12
1.2.1.1 <i>Objetivos Específicos</i>	12
2 FUNDAMENTAÇÃO TEÓRICA	13
2.1 Agentes e Sistemas Multiagentes	13
2.1.1 Agentes	13
2.1.1.1 <i>Categoria de Agentes</i>	16
2.1.1.2 <i>Aplicação de Agentes Inteligentes</i>	17
2.1.2 Sistemas Multiagentes	18
2.1.2.1 <i>Desafios e dificuldades de um Sistema Multiagente</i>	21
2.1.2.2 <i>Comunicação entre agentes</i>	22
2.1.2.3 <i>Coordenação entre agentes</i>	24
2.1.2.4 <i>Interação entre agentes</i>	26
2.2 Futebol de Robôs	27
3 ROBOCUP – “The Robot World Cup”	30
3.1 Apresentação	30
3.2 A UFSC e a RoboCup	30
3.3 Categorias	31
3.3.1 Robôs Pequenos – (Small size F180)	31
3.3.2 Robôs Pequenos – (Small size Full Set)	32
3.3.3 Robôs Médios – (Middle size)	32
3.3.4 Liga Simulador – (Simulador League)	32
3.3.5 Robôs Sony – (Robôs com pernas)	33
3.4 Pré-requisitos	33
3.4.1 Hardware e Softwares necessários	33

3.4.2	Protocolo de Comunicação	34
3.4.3	Problemas Computacionais	35
3.4.3.1	<i>Restrições Temporais</i>	35
3.4.3.2	<i>Avaliação das ações</i>	35
3.4.3.3	<i>Dinamismo do Ambiente</i>	36
4	SIMULADOR ROBOCUP	37
4.1	Apresentação	37
4.2	O Servidor	37
4.3	O Monitor	38
4.4	O LogPlayer	39
4.5	Informações controladas pelo Simulador	39
4.5.1	Campo	40
4.5.2	Colisões	40
4.5.3	Ruído e Vento	40
4.6	Regras do Jogo	40
4.6.1	Regras julgadas pelo árbitro automático	41
4.6.1.1	<i>Saída de bola</i>	41
4.6.1.2	<i>Gols</i>	41
4.6.1.3	<i>Bola fora do campo</i>	41
4.6.1.4	<i>Distância da bola</i>	42
4.6.1.5	<i>Modo de jogo</i>	42
4.6.1.6	<i>Meio e fim de jogo</i>	42
4.6.1	Regras julgadas pelo árbitro humano	42
5	SOCCER SERVER	44
5.1	Fazendo download	44
5.2	Apresentação	44
5.3	Objetos	45
5.4	Protocolos	46
5.4.1	Comandos do Protocolo Cliente	46
5.4.2	Controle do Cliente	48
5.4.3	Protocolo de Sensores do Cliente	49
5.5	Sensores	50
5.5.1	Sensor Auditivo	51
5.5.2	Sensor Visual	52
5.5.3	Sensor Corporal	54

5.6 Modelo de Ações	55
5.6.1 Agarrar (“Catch”)	55
5.6.2 Aceleração (“Dash”).....	56
5.6.2.1 Modelo de força ou energia (“Stamina”)	57
5.6.3 Chutar (“Kick”)	57
5.6.4 Movimentar-se (“Move”)	58
5.6.5 Dizer (“Say”)	59
5.6.6 Girar (“Turn”)	60
5.6.7 Girar o pescoço (“TurnNeck”)	60
5.7 Jogadores Heterogêneos	61
5.8 Parâmetros do Soccer Server	61
6 SOCCER MONITOR	64
6.1 Fazendo download	64
6.2 Apresentação	64
6.3 Sistemas de coordenadas	66
6.4 Tempo, Contagem de ciclos e Troca de Dados	66
6.5 As bandeiras de indicação no campo (Flags)	66
6.6 Executando o Monitor	67
7 SOCCER CLIENTE	70
7.1 Apresentação	70
7.2 Biblioteca de Agentes	70
7.3 Características de um agente jogador	71
7.4 Arquitetura Clássica de Agente	71
7.4.1 Percepção	72
7.4.1.1 <i>Memória</i>	73
7.4.2 Raciocínio	74
7.4.3 Ação	75
7.4.4 Aprendizado	76
7.5 Arquitetura Krislet	76
7.5.1 Apresentação	76
7.5.1.1 <i>Classe Krislet</i>	77
7.5.1.2 <i>Classe Brain</i>	77
7.5.1.3 <i>Classe Memory</i>	78
7.5.1.4 <i>Classe VisualInfo</i>	78
7.5.1.5 <i>Outras Classes</i>	78
7.5.2 Desempenho	78

7.6 Estratégias de novas Arquiteturas mais elaboradas	79
7.6.1 Arquitetura Reativa	79
7.6.2 Arquitetura de Subordinação	79
7.6.2.1 <i>Cooperação sem Deliberação</i>	80
7.6.3 Arquiteturas Propostas	81
7.6.3.1 <i>Jogador Centro-Avante</i>	82
7.6.3.2 <i>Jogador Zagueiro</i>	83
7.7 Principais Comportamentos dos agentes jogadores	85
7.8 Comandos de Controle	89
7.8.1 Comandos Corporais	89
7.8.2 Comandos de Comunicação	90
7.8.3 Outros Comandos	91
8 O TREINADOR	92
8.1 Apresentação	92
8.2 Diferença entre o Treinador On-line e o Treinador de Arquibancada ..	92
9 CONSIDERAÇÕES FINAIS	94
10 REFERÊNCIAS BIBLIOGRÁFICAS	96
11 ANEXOS	98
11.1 Artigo sobre o Trabalho	99

LISTA DE TABELAS

Tabela 2.1 - Principais características dos agentes	15
Tabela 2.2 – Resumo das características envolvidas nos Sistemas Multiagentes	20
Tabela 2.3 - Tabela Comparativa entre uma partida de Xadrez e de Futebol de Robôs	29
Tabela 5.1 – Estabelecimento e Re-estabelecendo conexão e desconectando do servidor	46
Tabela 5.2 – Tabela com os Comandos de Controle do Cliente	48
Tabela 5.3 – Tabela com os Protocolos de Sensor do Cliente	49
Tabela 5.4 – Parâmetros do sensor auditivo	51
Tabela 5.5 – Parâmetros do sensor visual	53
Tabela 5.6 – Parâmetros do sensor corporal	54
Tabela 5.7 – Parâmetros da área de atuação do goleiro	56
Tabela 5.8 – Parâmetros básicos para ações de colisão (“dash”)	58
Tabela 5.9 – Parâmetros básicos para ações como chutar (“kick”) e a bola	59
Tabela 5.10 – Parâmetros básicos para ações como dizer (“say”)	59
Tabela 5.11– Parâmetros básicos para ação de girar (“turn”)	60
Tabela 5.12 – Parâmetros básicos para ação d girar o pescoço (“turnNeck”)	61
Tabela 5.13 – Parâmetros básicos do Simulador Soccer Server	61
Tabela 7.1 – Tabela com as características de um agente jogador	71
Tabela 7.2 – Representação das regras de comportamento para um agente zagueiro.....	84

LISTA DE ILUSTRAÇÕES

Figura 2.1 – Desenho ilustrativo do protocolo de negociação segundo VREE95	26
Figura 5.1 – Diagrama UML demonstrando a visão geral dos objetos dentro do simulador.....	45
Figura 5.2 – Representação das variáveis responsáveis pela visão de um jogador	53
Figura 5.3 – Representação da área de atuação de um goleiro	55
Figura 6.1 – Interface de um Modelo de Soccer Monitor	65
Figura 6.2 – Representação do campo marcado com bandeiras (flags)	67
Figura 6.3 – Janela de conexão entre o Soccer Monitor e o Soccer Server	68
Figura 7.1 – Modelo da Arquitetura Clássica de Agentes de Russ95	72
Figura 7.2 – Figura ilustrativa de como as mensagens são interpretadas	73
Figura 7.3 – Representação do ambiente sendo armazenado em memória pelo jogador	74
Figura 7.4 – Representação da Arquitetura do agente Krislet	76
Figura 7.5 – Representação do funcionamento do jogador Krislet	77
Figura 7.6 – Representando a ação de um jogador segundo a Arquitetura de Ludwig,Cordenonsi	83

RESUMO

PEREIRA, Jan Antonio. **Estudo Sistemático de Auxílio à implementação de um time de futebol de robôs simulados no Simulador RoboCup**. 2004 - 102 páginas - Monografia (Graduação em Bacharelado em Ciências da Computação)- Universidade Federal de Santa Catarina, Florianópolis, 2004.

O presente estudo propõem-se a recolher uma documentação que venha motivar pesquisadores na Área de Inteligência Artificial e Robótica através de um ambiente de simulação aberto a investigação e a descobertas de novas tecnologias que possivelmente venham a ser aplicadas em problemas sociais e industriais. Para realizar a referida pesquisa pautou-se de um abordagem qualitativa, que ofereceu sustentação a pesquisa-ação realizada, a qual utilizou-se da pesquisa bibliografia e descritiva. Para alcançar o objetivo proposto, inicialmente são apresentados os diversos conceitos de agentes e sistemas multiagentes, importantes para o contexto de uma partida de futebol de robôs. A seguir descreve-se o ambiente de Simulação Soccer Server, sua arquitetura cliente-servidor, seus pré-requisitos, forma de comunicação, comandos, parâmetros, regras do jogo, além de citar algumas propostas de arquiteturas de agentes jogadores de futebol. No decorrer do estudo foram utilizadas várias tabelas, quadros esclarecendo melhor os dados pesquisados. Como resultado pode-se esperar um estudo que reúne uma documentação que espera-se ser útil para um primeiro contato entre o pesquisador interessado e o ambiente de simulação Soccer Server da RoboCup.

PALAVRAS-CHAVES: Agentes; Sistemas Multiagentes; Simulador Soccer Server; RoboCup; Futebol de Robôs.

ABSTRACT

PEREIRA, Jan Antonio. **Estudo Sistemático de Auxílio à implementação de um time de futebol de robôs simulados no Simulador RoboCup.** 2004 - 102 páginas - Monografia (Graduação em Bacharelado em Ciências da Computação)- Universidade Federal de Santa Catarina, Florianópolis, 2004.

This study proposes to join a documentation that to motive searches in The Artificial Intelligence and Robotic through the open simulation environment, an investigation and discovery new technologies that possibility to apply in social and industrial problems. To realize this search, take quantitative approach that offered sustentation to make action-search, using biography and descriptive investigation. To reach proposed goal initialed showing various agents and mult-agents systems conceptions essential to the context of robot soccer game. After then, described simulation soccer server environment, client-server architecture, prerequisites, communication structure, commands, parameters, game roles, beside to cite some player soccer agent architecture. Through the study were many tables and squares to better elucidate the investigation data set. As result, can to hope that the study meet utilities documentations for the first contact in between researchers and Soccer Server Simulation Environment of RoboCup.

Key-words : Agent, Multi-agents Systems, Soccer Server Simulation, RoboCup, Robot Soccer Game.

1. INTRODUÇÃO

Na tentativa de estimular a área de Inteligência Artificial e de Sistemas Multiagentes (SMA), atraindo a atenção do público, cientistas japoneses desenvolveram em 1996 uma competição de robôs jogadores do esporte mais popular no mundo, o futebol, competição essa denominada Robocup, onde várias tecnologias podem ser testadas e aplicadas para a solução de um problema. A RoboCup rapidamente vêm atraindo ambos os lados acadêmicos e comerciais, intensificando a comunicação e negociação entre entidades inteligentes, raciocínio qualitativo e inteligência distribuída.

A utilização do Simulador Soccer Server da RoboCup, como ambiente de simulação de uma partida de futebol, permite a aplicação de diversos conceitos ligados à Inteligência Artificial tais como : redes neurais, técnicas de aquisição de conhecimento, computação evolutiva, teoria de grafos, técnicas de aprendizagem, agentes autônomos, sistemas multiagentes, entre outros assuntos.

Uma partida de futebol, ou uma simulação dela, como veremos a seguir, é um problema extremamente dinâmico, imprevisível e não-determinístico, o que torna a tarefa dos pesquisadores, não apenas desafiadora, mas também recompensadora e motivadora, fato que o aspecto competitivo do evento ajuda a estimular.

Dentro de um contexto geral, o presente trabalho no primeiro momento faz uma apresentação sobre o que seria um jogo de futebol de robôs, e aborda conceitos como agentes e sistema multiagentes fundamentais na elaboração de agentes jogadores de futebol. Em um segundo momento, faz um estudo sistemático sobre a RoboCup, apresentando seu simulador baseado na arquitetura cliente/servidor, fazendo o levantamento dos pré-requisitos, forma de comunicação, comandos, parâmetros do simulador, regras do jogo, além de citar algumas arquiteturas possíveis para o interessado no desenvolvimento de agentes jogadores de futebol.

1.1 JUSTIFICATIVA

Estimulado pela iniciativa da RoboCup e pelo fascínio que sempre tive sobre os assuntos que envolvem a Inteligência Artificial e a Robótica, após me deparar com materiais dispersos, em língua estrangeira e geralmente tratando sobre uma abordagem específica o

assunto, notou-se a carência por um material que reuni-se as informações relevantes para um primeiro contato com interessados em desenvolver uma equipe jogadora de futebol de robôs simulados pelo Simulador Soccer Server da RoboCup.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Fazer um estudo sistemático de auxílio à implementação de um time de futebol de robôs simulados no Simulador RoboCup.

Abrir novas perspectivas quanto a futuras pesquisas nas Áreas de Inteligência Artificial e Robótica, bem como potencializar o nível de aprendizado nessas áreas, através da utilização do Simulador Soccer Server como ambiente de investigação e descoberta de novas tecnologias.

Colocar o Departamento de Informática e Estatística (INE) da Universidade Federal de Santa Catarina, por dentro da iniciativa da RoboCup, dando o primeiro passo em direção a novas pesquisas sobre o tema;

1.2.2 Objetivos Específicos

Relacionar os diversos conceitos de agentes e sistemas multiagentes dentro do contexto de uma partida de futebol de robôs.

Apresentar o ambiente de Simulação Soccer Server sua arquitetura cliente/servidor, seus pré-requisitos, forma de comunicação, comandos, parâmetros, regras do jogo, além de citar algumas propostas de arquiteturas de agentes jogadores de futebol de robôs, sempre que possível fazendo uso de tabelas e figuras esclarecedoras.

2. FUNDAMENTAÇÃO TEÓRICA

2.1. Agentes e Sistemas Multigentes

2.1.1 Agentes

Encontrar um único conceito ou padrão para agentes é muito difícil, pois há várias abordagens do ponto de vista de diversos autores que pode gerar significados diferentes.

Segundo o dicionário (Bueno1996) agente é o que trata de negócio por conta alheia; causa; o promotor ; o que pratica ação; tudo o que opera;

Na definição do dicionário (LUFT 1998) um agente é uma pessoa que age por ou no lugar de outra segundo autoridade por ela outorgada - é um representante da pessoa.

Em inteligência artificial, pode ser usado o termo agente para referir-se a um processo ou tarefa computacional em que o homem constrói uma ou várias funções com a capacidade de resolver problemas, adaptando-se ao um meio, reagindo a ele e provocando mudanças neste meio. Ou seja, ser capaz de aprender e tomar decisões a partir de situações diferentes.

A FIPA(Foundation for Intelligent Physical Agents) define agente como uma entidade que reside em um ambiente onde a interpretação de dados através de sensores refletem o ambiente e a partir daí a entidade é capaz de executar ações que produzem efeitos no ambiente. Um agente pode ser implementado a nível de software ou hardware.

Alguns Pesquisadores Michael Wooldridge e Nick Jennings (WOOLDRIDGE 1995), adotaram duas definições gerais: noção fraca e noção forte de agentes. Na definição ou noção fraca de agentes, eles conceituam como sistemas computacionais, sendo hardware ou software, com certas propriedades tais como autonomia, habilidade social, reatividade e pró-atividade. Na noção forte de agentes, mais adotada pelos pesquisadores ligados a área de Inteligência Artificial, possui, além das propriedades acima citadas, noções relacionadas ao comportamento humano, tais como o conhecimento, a crença, a intenção e a obrigação.

FERBER 1999, classifica agente como sendo uma entidade física ou virtual. Física, seria alguma coisa concreta e que atue no mundo real. Virtual seriam justamente entidades abstratas tais como componentes de softwares.

DEMAZEAU 1995 definiu como agente todas as entidades ditas ativas de um sistema. Este conjunto de agentes forma uma sociedade. As entidades passivas são designadas pelo termo ambiente. Um agente recebe informações e raciocina sobre o ambiente e sobre outros agentes, decidindo quais ações deve realizar e quais objetivos deve seguir. Um agente é uma entidade ativa, ou seja capaz de controlar suas ações, diferentemente das noções estáticas tais como módulos, conjunto de regras e bases de conhecimento.

A interação de tais agentes autônomos no ambiente está associada a uma série de problemas. Por exemplo, a impossibilidade de descrever completamente o ambiente e os obstáculos que o agente irá encontrar; a necessidade de situações imprevisíveis levarem a replanejamentos e novas ações, a cooperação entre agentes dentre outros problemas a enfrentar.

Seguindo estes dados, agentes seriam entidades autônomas que atuam em determinados ambientes de forma a interagir com este e com outros agentes. Sem necessitar de intervenções humanas persistentes.

Dentro desse amplo campo, existem inúmeras definições para “agentes”. A seguir temos uma coleção de definições retiradas de Bianchi(1998):

- um agente é qualquer coisa que pode ser vista como percebendo seu ambiente através de sensores e agindo sobre este ambiente através de efetadores;
- agentes autônomos são sistemas computacionais que habitam algum ambiente dinâmico e complexo, percebem e atuam autonomamente neste ambiente e, fazendo isto, atingem um conjunto de objetivos ou tarefas para os quais foram projetados;
- um agente é definido como uma entidade de software persistente dedicada a um propósito específico;
- agentes inteligentes realizam continuamente três funções : percebem as condições dinâmicas em um ambiente; agem para afetar as condições do ambiente; e raciocinam para interpretar as percepções, resolver problemas, realizar inferências e determinar ações;

- um agente pode ser um sistema computacional baseado em hardware ou(mais habitualmente) em software que possui as seguintes : autonomia, habilidade social, reatividade e pró-atividade;
- agentes autônomos são sistemas capazes de ações autônomas e propositadas no mundo real.

A partir dessas definições, apesar de variadas, algumas características básicas que os agentes devem possuir podem ser definidas. Algumas das mais importantes são:

PROPRIEDADE	SIGNIFICADO
Autônomo	Controle sobre as próprias ações
Pró-atividade (orientado a objetivos)	Segue objetivos e não apenas reage a mudanças no ambiente
Reativo	Respostas assíncronas a mudanças no ambiente
Temporalmente Contínuo	E um processo executado continuamente
Comunicativo	Comunica-se com outros agentes, incluindo usuários
Aprendizado	Muda seu comportamento baseado em experiências anteriores
Móvel	Pode transportar a si próprio de uma máquina a outra
Flexível	Conjunto de ações não é pré-definido
Personalidade	Personalidade verossímil e estado emocional

Tabela 2. 1 - Principais características dos agentes

Algumas propriedades são essenciais para uma melhor caracterização de agente. Uma delas é ser autônomo. É a autonomia que leva o agente inteligente a tomar decisões importantes para a conclusão de uma tarefa ou objetivo sem a necessidade da interferência do ser humano ou qualquer entidade. Ser capaz de agir independentemente com seu ambiente através de seus próprios sensores ou com as suas próprias percepções como objetivo de realizar alguma tarefa seja ela externa ou gerada por ele próprio.

Ligado a autonomia esta a pró-atividade, que nada mais é que a capacidade que o agente deve ter de tomar iniciativas. Eles não responde simplesmente de acordo com o meio. Têm a capacidade de mostrar comportamentos baseados em objetivos.

Reatividade é a capacidade de reagir rapidamente a modificações no ambiente, ou seja, perceber o meio e responder de modo conveniente.

As primeiras três características são o mínimo necessário para que um programa possa classificar-se como agente. As restantes dizem respeito a facilidades que podem ou não estar presentes, dependendo principalmente do ambiente de atuação do agente em ambientes complexos, como o mundo físico, exigem agentes complexos e a presença ou não de outros agentes com os quais devem se comunicar. Em particular, os requisitos da flexibilidade e aprendizado são definitivamente uma exigência para que um agente possa rotular-se inteligente. Finalmente, as questões relativas a personalidade são relevantes principalmente quando é desejável interagir com pessoas num ambiente normalmente humano, tipicamente sistemas de chat online (como ChatterBot), ou mesmo jogos de computador, criando personagens mais verossímeis (FONER 93).

Embora muitos agentes desenvolvam atividades sob a instrução de pessoas (por exemplo, uma busca na web por preços mais baixos para um determinado produto), muitos desenvolvem suas ações baseados não em um pedido externo de um usuário, mas baseados em seu conjunto de regras e conhecimento. Esses agentes normalmente realizam tarefas de manutenção de sistemas, como atualizações de bases de dados, verificação de sistemas de arquivos ou monitoramento de recursos críticos. Também estariam encaixados nessa categoria os agentes jogadores da Robocup, que uma vez inseridos no seu ambiente de jogo, o simulador Soccer Server exerce a função para a qual foram criados sem qualquer intervenção externa, exceto talvez pelo juiz humano da partida, que não interage diretamente com os clientes mas sim com seu ambiente.

Contudo, um agente não precisa ter todas essas características ao mesmo tempo. Existem agentes que possuem algumas, outros que possuem todas, o que é certo é que atualmente existe pouca concordância sobre a importância dessas propriedades e se é necessária sua obrigatoriedade para a caracterização de um agente. O consenso é que essas características tornam em muito um agente diferente de simples programas e objetos.

2.1.1.1 Categoria de Agentes

Existem diversos tipos de agentes e suas mais variadas diferenças na sua empregabilidade. É possível fazer uma classificação de agentes de acordo com vários

aspectos quanto à mobilidade, quanto ao relacionamento interagentes e quanto à capacidade de raciocínio.

Agentes móveis - São agentes que tem a mobilidade como característica principal. Capacidade de mover-se seja por uma rede interna local(intranet) ou até mesmo pela Web, transportando-se pelas plataformas levando dados e códigos.

Agentes situados ou estacionários – São fixos em um mesmo ambiente ou plataforma. Não se movimentam em uma rede e muito menos na Web.

Agentes competitivos - São agentes que competem entre si para a realização de seus objetivos ou tarefas. Ou seja, não há colaboração entre os agentes.

Agentes coordenados ou colaborativos – São agentes com a finalidade de alcançar um objetivo maior, realizam tarefas específicas coordenando-as entre si de forma que suas atividades se completem.

Agentes Reativos - É um agente que reage a estímulos sem ter memória do que já foi realizado no passado e nem previsão da ação a ser tomada no futuro. Não tem representação do seu ambiente ou de outros agentes e são incapazes de prever e antecipar ações. A força de um agente reativo vem da capacidade de formar um grupo e construir colônias adaptando-se a um ambiente.

Agentes Cognitivos – O agente cognitivo é capaz de resolver problemas por ele mesmo. Ele tem objetivos e planos explícitos os quais permitem atingir seu objetivo final. O agente é capaz de controlar o seu próprio comportamento

2.1.1.2 Aplicação de agentes inteligentes

Segue abaixo apenas das inúmeras aplicações de agentes inteligentes:

- Agentes no processo ensino-aprendizagem
Utilização de agentes no aprendizado e no processo de treinamento de usuários.
- Agentes na Indústria
- A grande vantagem da aplicação de agentes na produção industrial è o fato de que muitos dos sistemas que atuam são complexos e isolados. O papel dos agentes seria integrá-los e compartilhar informações entre os sistemas.

- Agentes em Simulação
Agentes podem simular situações dando um grau maior de veracidade. Tanto na área de entretenimento quanto a área de pesquisa tecnológica e militar.
- Agentes em realidade virtual

Nessas aplicações o agente atua como um participante que auxilia e monitora certas atividades e usuários, assistindo-os ou ajudando-os quando necessário, principalmente em ambiente virtuais muito complexos.

2.1.2 Sistemas Multiagentes

Sistemas Multiagentes caracteriza-se pela existência de agentes que interajam de forma autônoma e trabalhem juntos para resolver um determinado problema ou objetivo(TORSUN 1995).

Os sistemas Multiagentes permitem modelar o comportamento de um conjunto de entidades inteligentes e organizadas de acordo com leis do tipo social. Essas entidades, ou agentes, dispõem de uma certa autonomia e estão imersos num ambiente com o qual necessitam interagir, pelo que devem possuir uma representação parcial deste ambiente e meios de percepção e comunicação.

Segundo Bordini(2001), os Sistemas Multiagentes formam uma área de pesquisa dentro da Inteligência Artificial Distribuída(IAD), que se preocupa com todos os aspectos relativos à computação distribuída em sistemas de inteligência artificial. Em Sistemas Multiagentes, o enfoque principal é prover mecanismos para a criação de sistemas computacionais a partir de entidades de software autônomas, denominadas agentes, que interagem através de um ambiente compartilhado por todos os agentes de uma sociedade, e sobre o qual estes agentes atuam, alterando o seu estado.

Diante disto, quer-se dizer que é preciso prover mecanismos para a interação e coordenação destas entidades(agentes), já que cada uma possui um conjunto de capacidades específicas, bem como possuem seus próprios objetivos em relação aos estados do ambiente que querem atingir, exatamente porque cada agente possui um conjunto específico e

limitado de capacidades. Frequentemente os agentes precisam interagir para atingirem seus objetivos.

Em suma, pode-se dizer que Sistemas Multiagentes são sistemas constituídos de múltiplos agentes que interagem ou trabalham em conjunto de forma a realizar um determinado conjunto de tarefas ou objetivos. Esses objetivos podem ser comuns a todos os agentes ou não. Os agentes dentro de um sistema multiagente podem ser heterogêneos ou homogêneos, colaborativos ou competitivos, etc. Ou seja, a definição dos tipos de agentes depende da finalidade da aplicação que o sistema multiagente está inserido.

Os sistemas multiagentes com agentes reativos são constituídos por um grande número de agentes. Estes são bastante simples, não possuem inteligência ou representação de seu ambiente e interagem utilizando um comportamento de ação/reação. A inteligência surge a partir de várias trocas de interação entre os agentes e o ambiente. Ou seja, os agentes não são tão inteligentes individualmente, mas o comportamento global é.

Já os sistemas multiagentes constituídos por agentes cognitivos possuem uma quantidade bem menor de agentes cognitivos comparado aos sistemas multiagentes reativos. Estes, conforme a definição de agentes cognitivos, são inteligentes e contêm uma representação parcial de seu ambiente e dos outros agentes. Podem, portanto, comunicar-se entre si, negociar uma informação ou serviço e planejar uma ação futura. Esse planejamento de ações é possível pois em geral os agentes cognitivos são dotados de conhecimentos, competências, intenções e crenças, o que lhes permite coordenar suas ações visando a resolução de um problema ou a execução de um objetivo.

Atualmente a pesquisa em sistemas multiagentes está interessada principalmente na coordenação das ações e comportamentos dos agentes, como eles coordenam seu conhecimento, planos, objetivos e crenças com objetivo de tomar ações ou resolver problemas.

CARACTERÍSTICAS	PROPRIEDADE	VALORES POSSÍVEIS
Intrínsecas Do agente	Tempo de duração	De transiente a de vida longa
	Nível Cognitivo	De reativo a deliberativo
	Construção	De declarativo a procedimental
	Mobilidade	Estacionário a itinerante
	Adaptabilidade	Fixa –lecionável- autodidata
	Modelagem	Do ambiente, dele próprio ou de outros agentes
Extrínsecas Do agente	Localidade	De local a remoto
	Autonomia social	De independente a controlado
	Sociabilidade	Autista,ciente, responsável, membro de um time
	Amabilidade	Cooperativo- competitivo- antagonista
	Interações	Logística: direta ou com facilitadores, nível semântico: declarativas ou procedimentais
Do Sistema (sociedade de agentes)	Unicidade	De homogêneo a heterogêneo
	Gtularidade	Defina a grossa
	Estrutura de controle	Hierárquica a democrática
	Autonomia de interface	Específica comunicação – intelecto – habilidades
	Autonomia de execução	Independente ou controlado
Do Framework	Autonomia de projeto	Plataforma – linguagem – arquitetura- protocolo de interação
	Infra- estrutura de comunicação	Memória compartilhada ou baseado em mensagens; Conectado ou não; Ponto-a-ponto- multicast – broadcast; Push ou oull; Síncrono ou assíncrono
	Serviço de mediação	Baseado em ontologias; transacional
	Protocolo de mensagens	KQML;HTTP e HTML; OLE; CORBA; DSOM
Do Ambiente	Conhecimento	Quanto o agente conhece do ambiente?
	Previsibilidade	Quanto o agente pode prever sobre o ambiente?
	Controlabilidade	Quanto o agente pode controlar o ambiente?
	Historicidade	Estados futuros dependem de estados passados?
	Teleologicidade	Outras partes do ambiente possuem propósito? (i.e. existem outros agentes?)
	Tempo real	O ambiente se modifica enquanto o agente delibera?

TABELA 2. 2 – Resumo das Características envolvidas nos Sistemas Multiagentes

2.1.2.1 Desafios e dificuldades de um Sistema Multiagente

Segundo JENNINGS 1998, algumas razões levam o crescimento do interesse em pesquisas com sistemas multiagentes, são eles:

- A capacidade de fornecer robustez e eficiência.
- A capacidade de permitir interoperabilidade entre sistemas legados.
- A capacidade de resolver problemas cujo dado, especialidade ou controle é distribuído.

Apesar de muitas vantagens, sistemas multiagentes ainda possuem muitos desafios e dificuldades tanto em relação ao projeto como implementação. Segue alguns exemplos abaixo:

- Como programar, descrever, decompor e alocar problemas e sintetizar resultados com um grupo de agentes inteligentes?
- Como permitir a comunicação e a interação entre agentes? Quais as linguagens de comunicação e protocolos usar? O que e quando comunicar?
- Como assegurar que agentes coerentemente tomem decisões ou executem ações?
- Como permitir agentes individuais atuar e raciocinar sobre ações, planos e conhecimento sobre outros agentes como coordená-los? Como raciocinar sobre o estado de seus processos coordenados?
- Como identificar e reconciliar pontos de vistas diferentes e intenções conflitantes em uma coleção de agentes que tenham que coordenar suas ações?
- Como equilibrar efetivamente computação local e comunicação? Mais genericamente, como gerenciar alocação de recursos limitados?
- Como evitar ou minimizar comportamento prejudicial do sistema, tal como um comportamento caótico?
- Como projetar e construir sistemas multiagentes práticos? Como projetar plataformas tecnológicas e metodologias de desenvolvimento para sistema multiagentes?

Analisando todos , podemos resumir em apenas três principais desafios que um sistema multiagente possui. Em primeiro lugar diz respeito à comunicação. Como ela seria realizada entre os agentes e que tipo de protocolos usar? A segunda seria a interação. Como essa interação ocorrerá? Que linguagem os agentes devem usar para interagirem entre si e combinar seus esforços? E por último a coordenação. Como garantir coordenação entre os agentes para que haja uma coerência na solução do problema ou objetivo que estão tentando resolver ?

2.1.2.2 Comunicação entre agentes

A comunicação é fundamental para permitir que haja colaboração, negociação, e cooperação entre entidades independentes. Fornece a base necessária para a realização da cooperação entre múltiplos agentes.

A comunicação entre os vários agentes resolvedores de problemas permite a exploração em comum dos seus recursos e conhecimentos próprios, tornando possível o trabalho em paralelo de diferentes partes do problema, e a obtenção mais rápida da resolução do problema. Em sistemas multiagentes, é necessário que a comunicação seja disciplinada para que os objetivos sejam alcançados efetiva e eficientemente, necessitando assim uma linguagem que possa ser entendida pelos outros agentes presentes no ambiente. Essa comunicação tem como principal objetivo à partilha do conhecimento com os outros agentes e coordenação de atividades entre agentes. Ou seja, ela deve permitir que agentes troque informações entre si e coordenem suas próprias atividades resultando sempre em um sistema coerente.

Em sistemas onde não existe comunicação, denominados sistemas discretos, cada agente desempenha suas atividades independentemente um do outro, onde adquirem consciência de suas interações apenas através dos resultados das ações de seus companheiros sobre o ambiente. No oposto temos agentes totalmente conectados , onde um agente pode falar com qualquer outro.

Existem diversas maneiras de agentes trocar informações uns com os outros em sistemas multiagentes. A abordagem ideal para a comunicação entre agentes num ambiente multiagentes depende das características do sistema. Num sistema com agentes

heterogêneos, esforços como KQML(May95) e outros são indispensáveis, dado que a interoperabilidade entre esses agentes depende da utilização de um protocolo padrão de comunicação. KQML e outros trabalhos semelhantes procuram fornecer uma linguagem universal, com a qual os agentes desenvolvidos por qualquer pesquisador possa interagir com outros agentes sem nenhum conhecimento prévio sobre eles.

Segundo BAKER 1997, agentes podem trocar mensagens diretamente, chamada também por alguns autores como comunicação direta, podem comunicar-se através de um agente “facilitador” especial em sistema “federado”(comunicação assistida), para trabalhar em ambientes de rede, com agentes móveis ou estáticos, oferecendo serviços como roteamento de mensagens ou seja, comunicação por difusão de mensagens (“broadcasts”) e até utilizar o modelo de comunicação através de “blackboard” ou quadro-negro.

Comunicação direta, ou comunicação via troca de mensagem direta, cada agente comunica diretamente com qualquer outro agente sem qualquer intermediário. Nesse tipo de comunicação faz-se necessário que cada agente envolvido tenha conhecimento da existência dos outros agentes e da forma de como endereçar mensagens para eles. A principal vantagem deste tipo de comunicação entre agentes é o fato de não existir um agente coordenador da comunicação. As desvantagens são o custo da comunicação que se torna grande, principalmente quando há um grande número de agentes no sistema, e a própria implementação que se torna muito complexa em comparação às outras formas de comunicação.

A comunicação assistida (BAKER 1997), os agentes utilizam algum sistema ou agente especial para coordenar suas atividades. Ou seja, uma estrutura hierárquica de agentes é definida e a troca de mensagens dá-se através de agentes especiais designados facilitadores ou mediadores. Esse é uma alternativa bem popular à comunicação direta pois diminui o custo e a complexidade necessária aos agentes individuais na realização da comunicação.

Comunicação por quadro-negro ou “blackboard”, segundo BAKER 1997, é bastante usada na Inteligência Artificial como modelo de memória compartilhada. Ou seja, nada mais é que um repositório onde os agentes escrevem mensagens a outros agentes e obtêm informações sobre o ambiente. A comunicação entre os agentes é efetuada pela escrita e leitura da informação numa estrutura partilhada. Quando determinado item de informação,

presente no quadro-negro ou “blackboard” é importante para um agente, este lê essa informação e executa as ações que considera adequadas, de acordo com o seu conhecimento próprio.

Um sistema de quadro-negro ou “blackboard” é composto por três componentes básicos, segundo OLIVEIRA 97: A estrutura de dados “blackboard” ou simplesmente o “blackboard”, é uma base de dados global, partilhada pelas diferentes fontes de conhecimento, contém dados de entrada, soluções parciais e outros dados representativos do estado de resolução do problema. O “blackboard” serve também como um meio de comunicação e como um suporte para o mecanismo de ativação das fontes de conhecimento. Fontes de conhecimento (“Knowledge sources”), módulos independentes e separados de conhecimento aplicável aos possíveis estados do problema, mas que coletivamente contêm o conhecimento necessário para resolver o problema.

2.1.2.3 Coordenação entre agentes

Gerbard Weiss (WEISS 1999) afirma que “coordenação é uma característica fundamental para um sistema de agentes que executam alguma atividade em um ambiente.

A coordenação está muito relacionada com o compartilhamento do conhecimento entre os agentes, sendo, seu principal objetivo, tornar as ações individuais de cada agente coordenada para se atingir o objetivo final do sistema multiagente, Além disso, há uma preocupação com a coerência de modo a se discutir como o sistema multiagente por completo se comporta enquanto está resolvendo o problema. O principal motivo para uma preocupação maior com a coordenação entre agentes é o fato de que um só agente, dentro de um sistema multiagente, não terá informação ou capacidade suficiente para resolver muitos dos problemas , muitos dos objetivos não podem ser atingidos por agentes agindo isoladamente.

Uma vez comunicando-se, os agentes necessitam de uma maneira de coordenar suas ações para atingir seu objetivo. Essa coordenação deve levar em consideração tanto as funções de cada agente isolado, como o resultado combinado de todos eles operando simultaneamente. Embora possa não parecer um problema tão complexo numa primeira vista, a previsão dos resultados dos atos de múltiplos agentes autônomos trabalhando simultaneamente sobre seu ambiente pode ser muito difícil de prever.

Assim, coordenação seria a capacidade de esses agentes trabalharem em conjunto e combinar seus objetivos de forma a concluírem o objetivo final do sistema. Geralmente para uma cooperação ser bem sucedida, cada agente deve manter um modelo dos outros agentes e também desenvolver um modelo de interações futuras ou possíveis. Ela pode ser dividida em cooperação e negociação.

Quando os agentes executam uma política de cooperação, esta é o resultado da interação. Assim, o agente executa uma ação ou assume qualquer decisão como consequência da influência gerada pela presença ou conhecimento de outro agente.

A cooperação está fortemente ligada à aplicação específica, pois a sua base de trabalho implica a procura coletiva de um objetivo comum(resolução de um problema). A forma como a cooperação se desenrola depende, de como o problema a resolver é decomposto e distribuído.

Num sistema multiagente e seguindo um dos modelos possíveis Archon(WITTIG 92), é possível distinguir duas formas de cooperação quanto à partilha da informação:

- Partilha de resultados: Os agentes partilham entre si resultados parciais que são gerados com base em perspectivas possivelmente diferentes de uma tarefa global.
- Partilha de Tarefas: Acontece quando um agente detecta que não tem informação ou capacidades suficientes para executar uma dada tarefa. Para isso é necessário ter conhecimentos sobre os outros agentes.

De acordo com as relações de interdependência existentes, a cooperação pode ser classificada em quatro tipos :

- Cooperação Horizontal: Os agentes não são dependentes de qualquer outro agente do sistema. Porém, o uso de informação de outro agente, pode aumentar o grau de confiança que estes agentes receptores atribuem as suas soluções.
- Cooperação em árvore: Os agentes dependem de outros agentes do sistema para resolver os seus próprios problemas.
- Cooperação recursiva: Vários agentes dependem um dos outros para resolver problemas.

- Cooperação híbrida: Verifica-se quando ocorre cooperação horizontal inserida em cooperação recursiva ou em árvore.

O problema da negociação está mais ligado a interação entre agentes que representam interesses divergentes, e trata de como fazer com que um agente possa convencer outros a resolver situações de sua maneira.

Segundo WEISS 99, “Negociação é a coordenação entre agentes antagônicos ou simplesmente egoístas(self-interested)”. Ou seja, a negociação está relacionada à coordenação de agentes competitivos. Geralmente são usados protocolos de negociação para determinar regras de negociação e são definidos os conjuntos de atributos sobre os quais se pretende chegar a um acordo.

Em VREE 95, o protocolo utilizado durante a negociação deve ser também objeto de negociação, uma vez que depende do objeto da discussão. Existe uma ausência de teorias formais na área de comunicação e interação, o que torna o processo de negociação sujeito à instabilidade nas camadas inferiores.

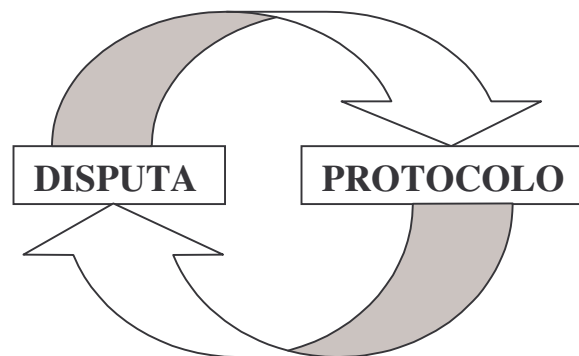


FIGURA 2.1 - Desenho ilustrativo do protocolo de Negociação segundo VREE 95

2.1.2.4 Interação entre agentes

O termo interação no contexto dos Sistemas Multiagentes significa a existência de qualquer tipo de ação coletiva no sistema. Ou seja, a interação com outros agentes implica que cada agente passa a ter conhecimento das atividades dos outros agentes, onde os objetivos globais podem levar a ações de cooperação ou competição.

A forma de interação que ocorre entre os agentes é um fator muito importante na integração destes. As linguagens de comunicação e sua expressividade definem a capacidade de comunicação de cada agente.

Com a preocupação que esta linguagem seja universal e partilhada por todos os agentes, o modelo de comunicação usada é o da comunicação humana utilizando a teoria dos atos comunicativos (“Speech Act Theory”).

Segundo MENESES 2001, esta teoria utiliza o conceito de performativas para permitir conduzir suas ações.

GUDWIN 2003 define algumas características desta teoria:

- Derivada da análise lingüística da comunicação humana.
- Com uma linguagem, um falante de uma língua não somente efetua uma declaração, mas realiza uma ação.
- Mensagens são ações ou atos comunicativos.

Gudwin afirma que os atos comunicativos são interpretados a partir da mensagem do contexto e nem sempre esta interpretação é óbvia. Ou seja, os atos comunicativos são sujeitos a interpretações dúbias que podem ter significados diferentes de acordo com o ponto de vista.

Ex: Saia da minha frente! – Comando

Por favor, saia da minha frente. – Pedido

Você poderia sair da minha frente? – Pergunta

Eu gostaria que você saísse da minha frente. – Informação

Diante de situações acima citadas, se faz necessário sempre deixar explícito o ato comunicativo relacionado à mensagem na comunicação entre agentes.

2.2 Futebol de Robôs

O xadrez foi um dos primeiros jogos onde foi aplicada a inteligência artificial. O desenvolvimento de máquinas que pudessem jogar sem o auxílio humano começou em meados dos anos 60. Russos e Americanos promoviam confrontos entre seus engenhos e grandes jogadores para saber qual das duas potências tinha o controle sobre a tecnologia da computação na época.

No entanto, a máquina demorou a vencer o ser humano. Só em 1997, o super computador Deep Blue, da IBM, derrotou o campeão mundial Garry Kasparov. O computador usava inteligência artificial do tipo informação perfeita, ou seja, com um número limitado de possibilidades.

Dado que o problema de jogar xadrez foi quase resolvido, buscando um novo desafio, um grupo internacional de pesquisadores em Inteligência Artificial e Robótica propôs um novo problema a ser solucionado: uma partida de futebol de robôs (LCMI, 2000; RoboCup 2001).

O futebol sem dúvida é um dos esportes mais praticados em todo o mundo, despertando o interesse de diversas pessoas em virtude do dinamismo da partida, colaboração entre jogadores, além das regras do jogo. Utilizar uma simulação desse tipo de ambiente acaba exigindo do desenvolvedor criar jogadores com características criativas, capaz de tomar iniciativas e decisões como na realidade acontece em uma partida de futebol real.

Partidas de futebol de robôs, além de serem extremamente motivantes para possibilitar o surgimento de um espírito de ciência e tecnologia nas jovens gerações, constituem uma atividade que possibilita a realização de experimentos reais para o desenvolvimento de robôs que apresentam comportamento inteligente e que cooperam entre si para a execução de uma tarefa comum, formando assim um time. Uma partida de futebol de robôs é dinâmica e imprevisível, resultando num domínio bastante complexo, que exige o uso de sistemas com alto grau de autonomia atuando em malha fechada e em tempo real para solucioná-lo. Diversos tópicos específicos de pesquisa são oferecidos por este domínio, incluindo, entre outros : (i) completa integração entre percepção, ação e cognição num time de múltiplos agentes robóticos; (ii) definição de um conjunto de comportamentos reativos robustos para cada agente, isto é, cada agente deve ser capaz de realizar tanto ações individuais quanto colaborativa; (iii) percepção em tempo real, robusta e confiável, incluindo rastreamento de múltiplos objetos em movimento.

Este domínio de aplicação permite que diversas técnicas sejam testadas e, principalmente, comparadas. A construção de um time de futebol de robôs envolve a integração de diversas tecnologias, tais como: projeto de agentes autônomos, cooperação em sistemas multi-agentes, estratégias de aquisição de conhecimento, engenharia de sistemas de

tempo real, sistemas distribuídos, reconhecimento de padrões, aprendizagem, controle de processos, etc. (LCMI, 2000).

Características	Xadrez	Futebol de Robôs
Meio Ambiente	Estático	Dinâmico
Mudança de Estado	Turnos de tempos	Tempo Real
Acessibilidade de informações	Completo	Incompleto
Controle	Central	Distribuído

Tabela 2.3 : Tabela Comparativa entre uma partida de Xadrez e de Futebol de Robôs

Com a iniciativa de motivar a pesquisa e promover as áreas de Inteligência Artificial e Robótica, um grupo de cientistas japoneses desenvolveram em 1996 uma competição de robôs jogadores do esporte mais popular no mundo, o futebol, surgindo assim a RoboCup "The Robot World Cup", onde as partidas são disputadas entre jogadores artificiais.

3 . ROBOCUP - “The Robot World Cup”

3.1 Apresentação

A RoboCup é uma iniciativa internacional entre robôs ou simuladores que partiu da idéia de se criar uma Copa do Mundo de Robôs. Não por acaso, já que o futebol, é um dos esportes mais populares do mundo, o que ajuda a incentivar, comparar e divulgar métodos e trabalhos. Promovendo-se o desenvolvimento de tecnologias passíveis de serem aplicadas em problemas sociais e industriais.

A meta central da RoboCup é, por volta do ano de 2050, colocar em campo um time de futebol de robôs humanóides autônomos para enfrentar o time campeão da copa do mundo da FIFA segundo regulamentações da mesma.

3.2 A UFSC e a RoboCup

A Universidade Federal de Santa Catarina (UFSC) foi a primeira entidade brasileira a participar em 1998 da Robot World Cup (RoboCup) realizada em Paris, França.

A equipe liderada pelos professores Augusto César Pinto Loureiro da Costa, do Departamento de Engenharia Elétrica e Guilherme Bittencourt , do Departamento de Engenharia e Automação na época, coordenaram um projeto que tinha como idéia criar as condições necessárias para, em breve, participar de competições com robôs físicos, como a Primeira Copa Brasil de Futebol de Robôs realizada entre os dias 28 a 30 de abril , daquele mesmo ano , em São Paulo.

A UFSC foi a São Paulo mas não competiu . Luciano Rottava da Silva , 21 anos , quinta fase do curso de Engenharia, fez parte da equipe de pesquisa com simuladores, acompanhou o campeonato e trocou idéias com outros grupos de pesquisa presentes. Segundo a entrevista dele a reportagem ao site Universidade Aberta , “estamos a frente de outras equipes no que diz respeito ao desenvolvimento de estratégias de cooperação entre agentes”. Nenhuma das universidades que participaram do evento na época usou Inteligência Artificial. Para desenvolver estratégias de jogo, foram empregadas soluções clássicas de computação, com um conjunto de ações pré-definidas. Por exemplo, na saída de bola, “girar para a esquerda e chutar ao gol”.

O time contava com onze jogadores comandados, cada um, por um programa independente. Como ocorre numa partida de futebol, cada jogador tem limitações quanto a sua visão em campo, além da necessidade de comunicação de modo a cooperarem para se chegar ao objetivo comum o gol.

A equipe recebeu uma colaboração fundamental de uma empresa francesa ILOG que através de sua representante no Brasil, a Choose Technologies, cedeu a equipe de pesquisa uma ferramenta computacional Ilog Rules para o desenvolvimento do time.

Os projetos sobre a RoboCup iniciados pelo departamento de engenharia, hoje, estão praticamente parados, restaram apenas alguns alunos doutorandos que possuem na universidade um projeto que visa provar que mini-robôs adquiridos pela universidade são capazes de jogar futebol.

Este trabalho visa colocar o Departamento de Informática e Estatística (INE), por dentro do assunto, procurando incentivar novos trabalhos na área, deixando nas mãos do departamento um bom material de auxílio no desenvolvimento de agentes jogadores de futebol, segundo o Simulador da RoboCup.

3.3 Categorias

A RoboCup possui várias categorias: sendo que destas todas, exceto uma, envolvem o uso de robôs reais. Esta exceção consiste numa plataforma cliente-servidor onde os jogadores são programas que conectam-se a um servidor central, o SoccerServer [Corten99], onde toda a partida é simulada.

3.3.1 Robôs Pequenos – (Small size F180)

Os robôs pequenos tem dimensões que ocupam no máximo uma área de 180 cm², com um comprimento menor ou igual a 18 cm. O campo tem as mesmas dimensões de uma mesa de tênis de mesa oficial, mas é marcada com vários identificadores visuais para auxiliar no reconhecimento das imagens do campo. A visão pode ser global, através de uma câmera posicionada acima do campo, ou individual, feita por cada robô. Os times são de no máximo 5 robôs-jogadores que correm atrás de uma bola de golf (tradicional) laranja.

3.3.2 Robôs Pequenos – (Small size Full Set)

As regras para essa categoria são essencialmente as mesma da categoria anterior, com exceção da quantidade de jogadores, que pode chegar a onze, e às dimensões do campo, que são três vezes maiores, tanto na largura quanto no comprimento. Também é permitido o uso de visão externa global.

3.3.3 Robôs Médios – (Middle size)

Os times de robôs médios podem ter no máximo 5 jogadores, com uma base de área circular não superior a 500 mm de diâmetro, ou 450 mm se retangulares. Não há limites definidos quanto à altura dos jogadores. Nesta categoria a cor da bola é vermelha. O campo tem 8220 mm de comprimento por 4575 mm de largura, e são proibidos quaisquer mecanismos de posse de bola que possam evitar que adversários não consigam tomar a posse da bola, assim como qualquer sistema de visão global. Vale lembrar que não existem paredes e nem sensores externos para os jogadores se localizarem, ao invés disso orientam-se através das balizas azuis e amarelas ao redor do campo.

3.3.4 Liga Simulador – (Simulator League)

Nesta categoria o campo e os jogadores não existem fisicamente, mas são simulados através de programas. Os jogadores são programas, chamados clientes, que conectam-se a um servidor central rodando o software oficial de simulação da RoboCup, soccerserver. As percepções e ações dos jogadores são mapeadas em mensagens enviadas através do protocolo UDP/IP, num ambiente de rede. Essa arquitetura permite que os clientes sejam escritos em qualquer linguagem que suporte esse tipo de comunicação.

Esta última categoria permite que grupos de pesquisadores em Inteligência Artificial desenvolvam times através da implementação de agentes computacionais autônomos capazes de cooperar para disputar uma partida de futebol de robôs, sem se preocupar com a parte física da construção de robôs (LCMI, 2000).

3.3.5 Robôs Sony (Robôs com Pernas)

Nesta categoria os robôs tem de coordenar todas as suas quatro pernas para levar a bola para dentro do gol. O robô Sony joga em um campo de carpete verde com declive.

Em todas as categorias a comunicação é implementada pelos engenheiros de maneira livre com exceção da categoria simulador. Como a intenção do simulador é recriar com fidelidade o ambiente de jogo de uma partida de futebol, a comunicação entre os clientes é proibida, a não ser através do servidor, que impõe todas as restrições cabíveis. Assim como não podemos esperar que os jogadores mantenham longas conversas durante uma partida real, não podemos esperar que os clientes troquem grandes quantidades de informação trivialmente durante o desenrolar de um jogo.

3.4 Pré-requisitos

Antes de apresentarmos o Simulador RoboCup e suas funcionalidades , é necessário deixar claro ao interessado a configuração mínima, quanto ao hardware e aos softwares, equipamentos e ferramentas estas fundamentais para iniciar o estudo e desenvolvimento de equipes de futebol de robôs, segundo a RoboCup.

3.4.1 Hardware e Softwares necessários

Hardware:

- **Computador pessoal**, Pentium II ou III 650 Mhz ou superior , com 128 MB de memória RAM ou mais, nesse caso , nada impede de se usar um computador inferior, apenas haverá perda de desempenho tornando o jogo mais lento);
- **Placa de rede 10/100** , caso a simulação esteja em fase de desenvolvimento e por esse motivo for feita em um único computador, não há a necessidade da placa de rede .

Softwares:

- **Sistemas Operacionais recomendados** : SunOS 4.1.x, Solaris 2, DEC OSF/1, NEW-OS, Linux , Irix 5 e apesar de não oficial é possível utilizar X-Windows.
- **Soccer Server** – software servidor .
- **Soccer Monitor** – software monitor .

3.4.2 Protocolo de Comunicação

A comunicação entre processos de software tornou-se indispensável nos sistemas atuais. Um dos mecanismos mais utilizados atualmente para possibilitar comunicação entre aplicações é chamado socket.

Existem 2 modos de utilização de sockets : o modo orientado a conexão, que funciona sob o protocolo TCP (Transmission Control Protocol, ou protocolo de controle de transmissão), e o modo orientado a datagrama, que funciona sob o protocolo UDP (User Datagram Protocol, ou protocolo de datagrama de usuários). Os dois modos funcionam sobre o protocolo IP (Internet Protocol).

A grosso modo, um Protocolo de Comunicação é uma seqüência de dados que contém um significado, transmitido por um meio. No caso da RoboCup, o protocolo utilizado é o UDP, o meio é a rede de cabos e o interpretador são as placas de rede.

No caso do protocolo UDP, os dados transmitidos se classificam em dois tipos : dados binários e dados texto. Dados binários são dados caracteres numéricos, que não representam , inicialmente, qualquer significado para o usuário apenas uma seqüência de zeros e uns , e dados texto são dados contendo sentenças de texto, constituídas pelos caracteres que estão sendo transmitidos por esta rede .

O protocolo UDP é usado pelo Simulador RoboCup por ser de rápida transmissão (cerca de dez vezes mais rápido que o protocolo TCP). A desvantagem em relação ao TCP está quanto a garantia a respeito da chegada da informação a seu destino. Mas no caso do Simulador RoboCup, esta desvantagem não causa grandes problemas já que , mesmo jogadores de futebol no mundo real não correspondem “100 %” , as tarefas exigidas por eles.

3.4.3 Problemas Computacionais

Antes de entrarmos a fundo sobre tudo que envolve a RoboCup é bom deixar claro algumas problemas que poderam surgir para qualquer interessado que deseje criar um agente jogador de futebol para a RoboCup.

3.4.3.1 Restrições Temporais

Quando um agente é colocado em um ambiente , seu primeiro passo se resume em determinar o estado que se encontra. Essa informação é fornecida não apenas pelos seus conhecimentos, mas principalmente pela percepção que o agente possui do ambiente a sua volta. Essa informação é obtida através de sensores. Diante disso, os agentes necessitam reagir rapidamente às mudanças do ambiente, e por isso todo o ciclo deve ocorrer com rapidez para que a percepção não se torne “jumpy”, isto é, o agente não perca grande quantidades de eventos pela frequência com que consegue obter dados de seus sensores. Se os passos forem executados seguindo uma seqüência, muitas vezes, o agente implementa um processo separado que fica responsável pela percepção a cada ciclo enquanto as outras fases são executadas. Mesmo que todas as mudanças no ambiente sejam captadas, isso não necessariamente significa que o agente terá tempo necessário para responder a todas elas.

Um dos problemas que podem ocorrer seria se o agente entrar na fase de raciocínio com uma representação do ambiente, e durante seu cumprimento essa representação é alterada de alguma forma, o que efetivamente transgrediria o ciclo descrito acima.

3.4.3.2 Avaliação das Ações

Tendo a análise das condições do agente e do seu ambiente, e baseado no seu conjunto de regras, finalmente é tomada a decisão sobre que ação deverá ser efetuada a seguir. O agente atua no ambiente através dos efetadores que exercem um papel inverso aqueles dos sensores : transformar intenções do agente em ações efetivas sobre os elementos do ambiente.

Comparando que em jogos de tabuleiros clássicos, é bem mais fácil analisar se as jogadas foram bem sucedidas ou não. Já em uma partida de futebol é bem mais difícil, no sentido de analisar que um passe errado dado no início do jogo fez a equipe perder.

Essa talvez seja uma restrição ainda maior do que a temporal, pois exige que a avaliação seja feita de forma mais ampla possível, tornando-se uma tarefa quase impossível de ser avaliada. Sendo que um agente só ajusta sua base de conhecimento diante de uma avaliação.

3.4.3.3 Dinamismo do Ambiente

O ambiente de jogo simulado na Robocup, uma partida de futebol, é muitas ordens de magnitude mais dinâmico do que aqueles encontrados pela maioria dos agentes habituais. Cada jogador deve ter uma função bem definida no jogo, e deve executá-lo de forma a não prejudicar a equipe como um todo. Essa característica é reforçada no ambiente da Robocup pela própria característica do futebol ser um esporte coletivo. Os agentes jogadores necessitam de mecanismos de comunicação, sincronização e cooperação para exercerem seus papéis.

4. SIMULADOR ROBOCUP

4.1 Apresentação

O Simulador da RoboCup é um sistema que simula o ambiente físico de uma partida de futebol e que permite aos jogadores virtuais competirem num ambiente multiagente incerto, complexo e dinâmico. O Simulador tenta disponibilizar um ambiente competitivo para a investigação em inteligência artificial, deixando a cargo dos investigadores o desenvolvimento das ações que simule seus agentes jogadores .

O Simulador RoboCup , está dividido em diversos componentes entre eles : o Servidor, o(s) Monitor(es), o(s) Cliente(s) ou melhor dizendo agentes jogadores , ferramentas de Log , entre outros .

4.2 O Servidor

O Servidor é um sistema que permite que várias equipes competirem em um jogo de futebol. Desde que seja realizado sobre o modelo Cliente / Servidor, não há nenhuma limitação a respeito de como as equipes são construídas. Todo cliente só envia mensagem ao servidor. Lembrando que toda a comunicação entre o servidor e cada cliente é estabelecida através de sockets UDP/IP. Cada cliente é um processo separado e conecta-se ao servidor através de uma porta específica. Depois que um jogador conecta ao servidor, todas as mensagens são transferidas através desta porta.

Uma equipe pode ter até 12 clientes, isto é 11 jogadores (10 jogadores + 1 goleiro) além de um técnico. Os jogadores emitem requisições ao servidor a respeito das ações que querem executar (por exemplo chute a bola, gira, avança, etc.). O servidor recebe essas mensagens, processa as requisições, e atualiza o ambiente de interação. Além disso, o servidor fornece a todos os jogadores informações sensoriais (por exemplo dados visuais a respeito da posição dos objetos no campo, ou dos dados sobre companheiros em campo, força ou velocidade). É importante mencionar que o servidor é um sistema em tempo real que trabalha com intervalos discretos de tempo (ou ciclos).

Cada ciclo tem uma duração específica, e as ações necessitam ser executadas em um dado ciclo, devem chegar ao servidor durante este intervalo. Conseqüentemente, um mal desempenho de um jogador resulta nas oportunidades de realizações de suas ações o que impactua no desempenho da equipe como um todo. Mais informações sobre o servidor, veja o capítulo 5 .

4.3 O Monitor

O monitor do jogo é uma ferramenta de visualização que permite ao técnico acompanhar o que está acontecendo dentro do servidor durante um jogo. Atualmente existem dois tipos de monitor, o rcssmonitor e o rcssmonitor_classic. A informação mostrada em ambos os monitores inclui o placar do jogo, os nomes das equipes, e as posições de todos os jogadores e da bola em campo.

Fornecem também relações simples ao Servidor. Como por exemplo, quando ambas as equipes conectam-se , a tecla "Kick-On" no monitor permite que um árbitro humano comece o jogo. O rcssmonitor, é baseado no frameview de Artur Merke, estende funcionalidades do monitor clássico por diversas características.

- É possível dar um Zoom em áreas do Campo. Isto é útil especialmente para eliminar erros.
- As posições correntes e as velocidades de todos os jogadores e da bola podem ser impressas no console.
- Uma variedade de informação pode ser mostrada no monitor, por exemplo a visão do jogador, sua força ou (no exemplo de jogadores heterogêneos) um tipo de jogador.
- Os jogadores e a bola podem quando necessário ser movidos com o mouse.

Existe ainda a possibilidade de se conectar vários monitores ao Servidor ao mesmo tempo (no caso de se mostrar o mesmo jogo em diversos terminais). Mais informações sobre o monitor veja o Capítulo 6.

4.4 O Logplayer

O Logplayer pode ser comparado como uma gravação do jogo. É uma ferramenta usada na repetição de lances de um jogo. Ao funcionar o servidor, determina as opções que foram parte do arquivo de log e armazena todos os dados em disco.

Então, o programa rcsslogplayer combinado com um monitor pode ser usado para rever um jogo tantas vezes quanto necessárias. Isto é completamente útil para fazer a análise de uma equipe e descobrir os pontos fortes ou fracos de uma equipe. O logplayer funciona como um vídeo é possível avançar ou retroceder lances a medida do desejado. Também o logplayer permite que você salte a um ciclo particular em um jogo (como por exemplo se você quiser somente ver os gols).

Finalmente o logplayer permite que você edite gravações existentes, isto é você pode conservar cenas interessantes de um jogo (ou de diversos jogos) a um outro arquivo de log e assim criar uma apresentação facilmente. O logplayer pode ser controlado através de uma GUI pequena ou de uma linha de comando. Além dos comandos que podem ser lidos de arquivos, nos quais adiciona potencialidades a scripts limitadas pelo logplayer.

4.5 Informações controladas pelo Simulador

O objetivo do simulador , é sem dúvida se aproximar o máximo possível da realidade de uma partida de futebol, para isso foi adicionado ao simulador diversas características que veremos a seguir :

4.5.1 Campo

O Campo de jogo e todos os objetos nele são bidimensionais. Não há nenhuma noção da altura de nenhum objeto. O tamanho do campo segue as regras do futebol humano, com um comprimento de 105 metros e uma largura de 68 metros . A largura do gol é de 14,64 metros, além disso jogadores e bola são representados por círculos em um ambiente chamado Soccer Monitor.

4.5.2 Colisões

Se durante um dado ciclo da simulação, dois objetos se sobrepuserem, os objetos são movidos para trás até que não se sobreponham. As velocidades são multiplicadas então pela nota -0.1 para evitar que a bola e jogadores se atravessem, obedecendo com isso a lei da física que diz que dois corpos não podem ocupar o mesmo espaço num mesmo instante de tempo.

4.5.3 Ruído e vento

A fim refletir movimentos inesperados dos objetos no mundo real, o Soccer Server adiciona ruídos e vento ao movimento dos objetos e aos parâmetros dos comandos.

4.6 Regras do Jogo

Como nos jogos reais de futebol, uma partida simulada também tem a necessidade de se reger de regras. Para detectar as infrações cometidas durante um jogo e tomar as respectivas atitudes em conformidade com o estipulado nas regras, existem dois árbitros, o árbitro automático e o árbitro humano. Cada um possui um conjunto de regras , que no decorrer da partida precisam identificar e classificar as infrações, punindo-as quando necessário, ou advertindo tal jogador infrator fazendo com isso, valer as regras do jogo.

4.6.1. Regras julgadas pelo árbitro automático

4.6.1.1 Saída de bola

Antes da saída de bola, quer seja no início do jogo, ou apitando um gol, os jogadores tem que se posicionar na sua metade do campo.

Para isso, após um gol, o árbitro dá 5 segundos para que os jogadores se posicionem na sua metade do campo. Os jogadores têm disponível para isso, o comando move, que lhes permite que se desloquem para um dado ponto do campo instantaneamente, evitando assim que eles tenham de correr (mais lento) e que gastem seu atributo força. Caso esta regra seja quebrada, o jogador em questão é colocado numa posição aleatória do seu lado do campo.

4.6.1.2 Gols

Quando um time marca um gol, o árbitro envia uma mensagem a todos os jogadores avisando que houve um gol na partida, atualiza o resultado, coloca a bola no centro e muda o modo do jogo para kick of <timeA ou timeB> (representando o time que deve iniciar a partida no centro de campo).

4.6.1.3 Bola fora do campo

Quando a bola sai fora do campo o árbitro posiciona a bola no lugar apropriado (na linha lateral, na linha de fundo ou na área do goleiro) e muda o modo de jogo para kick_in (lateral) , corner_kick (escanteio) ou goal_kick (tiro de meta), dependendo do lugar onde a bola saiu e de quem chutou. Se for um escanteio a bola é posicionada a 1 metro da linha lateral e a 1 metro da linha de fundo (correspondendo ao quarto de circunferência que existe nos cantos das linhas de fundo de um campo).

4.6.1.4 Distância da bola

Quando o modo de jogo é `kick_in` , `corner_kick` ou `goal_kick` o árbitro verifica se os jogadores que estão a defender se encontram na distância regulamentada. Para os jogadores estarem numa posição correta devem estar 9,15 metros da bola. Os jogadores removidos são colocados a volta dessa circunferência . Quando o modo de jogo é `off_side` , todos os jogadores atacantes são posicionados num local em que não estejam fora de jogo. São considerados atacantes os jogadores que estão fora do jogo e os que estão a 9,15 da bola.

Quando o modo de jogo é `goal_kick` todos os jogadores atacantes têm de sair da área de pênalti, e não podem entrar lá enquanto o pênalti seja cobrado. Após a marcação do pênalti e sua cobrança o modo de jogo é alterado para `play_on` (modo de jogo normal).

4.6.1.5 Modo de jogo

Quando o modo de jogo é `kick_in`, `corner_kick` ou `goal_kick` o árbitro o altera imediatamente para `play_on`, logo que a bola comece a andar.

4.6.1.6 Meio e fim de jogo

O árbitro suspende o jogo quando é atingido o final do primeiro e do segundo tempo de partida. Cada um dos tempos tem a duração de 3000 ciclos ou aproximadamente 5 minutos, se o jogo chega ao fim e as duas equipas se encontrarem empatadas, é dada uma prorrogação a partida até que uma das equipas marque um gol, o chamado gol de ouro.

4.6.2 Regras julgadas pelo árbitro humano

Embora o árbitro automático detecte muitas infrações, continuam a existir outras que lhe são complicadas de detectar e que não podem ser esquecidas. Por exemplo, o caso das obstruções que é difícil de analisar, uma vez que depende dos objetivos do infrator (só é obstrução se um jogador decidir deliberadamente bloquear a passagem do adversário).

Para resolver estas situações o servidor está equipado com uma interface para que um árbitro humano que possa intervir no jogo quando necessário obedecendo os seguintes tópicos aprovados como infrações pela RoboCup :

- Utilizar jogadores para cercar a bola não permitindo a passagem dos adversários;
- Bloquear o gol com jogadores;
- Não colocar a bola em jogo após um dado número de ciclos;
- Bloquear intencionalmente os outros jogadores;
- Abusar do comando catch do goleiro (lançando a bola ao chão e apanhando-a várias vezes de forma a movimentar-se em segurança dentro da área).

Além destas regras básicas é verificado se um dado jogador está enviando mais do que 4 mensagens por ciclo de simulação ou se possui outro tipo de comportamento inapropriado.

5. SOCCER SERVER

5.1 Fazendo download

O Soccer Server possui diversas versões, encontra-se disponível para download no site <http://sserver.sourceforge.net> .

Clique sobre o tópico Downloads localizado na frame do lado esquerdo, automaticamente a frame do lado direito carregará a seção de downloads onde lá você encontrará THE ROBOCUP SOCCER SIMULATOR SERVER, clique sobre o link para fazer seu download.

O Link clicado o levará a uma nova página com diversas versões do Soccer Server, escolha uma e inicie seu download.

5.2 Apresentação

O servidor, Soccer Server, é o “cérebro do jogo” do Simulador RoboCup, permite a competição de jogadores virtuais em um ambiente complexo e dinâmico que é executado em tempo real.

O jogo ocorre sob um sistema cliente/servidor que simula os movimentos da bola e dos jogadores, serve de intermediário na comunicação entre clientes (por exemplo, através de comandos say e hear) além de estabelecer as regras do jogo .

O Soccer Server, foi desenvolvido em C e C++, por questões de desempenho , e roda sem problemas em várias plataformas Unix, como SunOs, Solaris e Linux , entre outras. Apesar de existir versões de servidores para plataformas X-Windows estas são não oficiais.

O servidor trabalha em tempo real com intervalos discretos de tempo (ou ciclos). Cada ciclo tem a duração de 100 milissegundos, e as ações necessitam ser executadas dentro desse ciclo, e para isso devem chegar ao servidor durante este intervalo.

5.3 Objetos

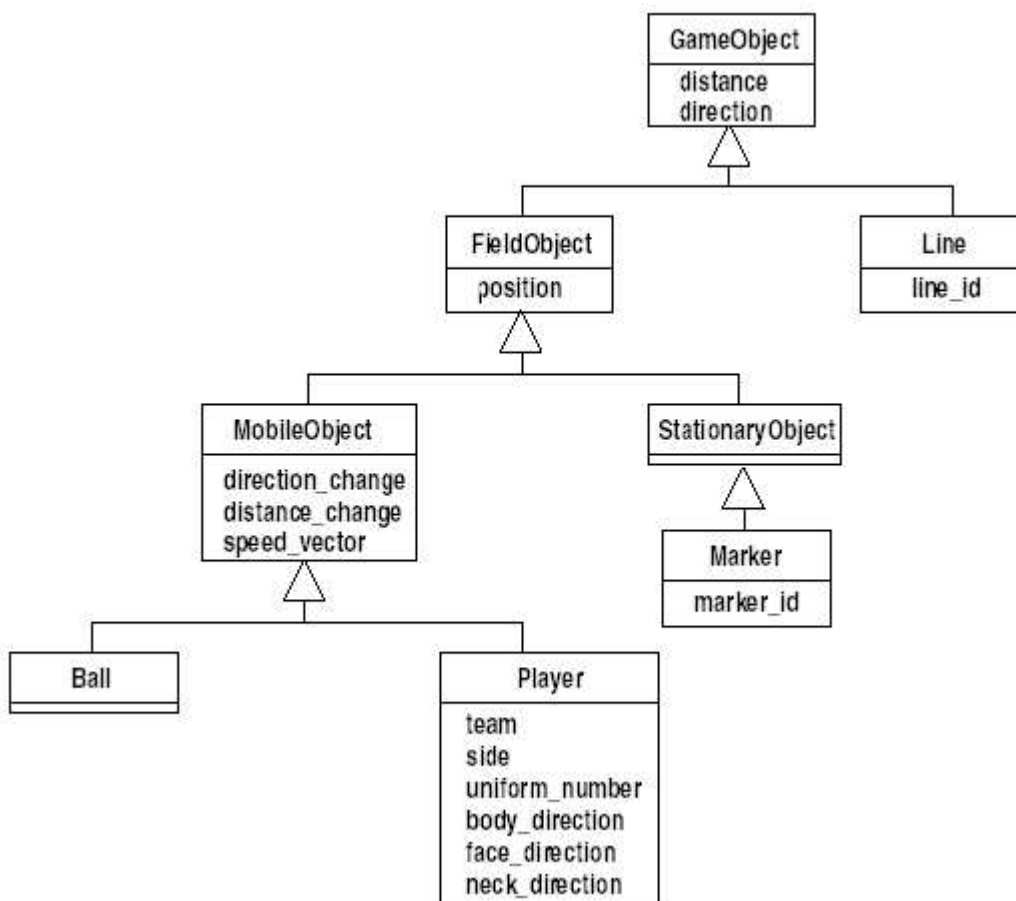


Figura 5.1 – Diagrama UML demonstrando a visão geral dos objetos dentro do simulador

Um Jogador (Player) herda as características de um objeto Móvel (MobileObject) que por sua vez herda as características do objeto Campo (FieldObject). A bola (Ball) herda, assim como o jogador as propriedades de um objeto Móvel (MobileObject).

Os pontos de orientação (Marker) herdam as características dos objetos estacionários (StationaryObjects), que, analogamente aos objetos Móveis também herdam as características do Campo. As linhas (Line) e o campo herdam as características do jogo (GameObject).

5.4 Protocolos

5.4.1 Comandos do Protocolo Cliente

Do Cliente para o Servidor	Do Servidor para o Cliente
<p>(init TeamName [(version VerNum)] [(goalie)])</p> <p>TeamName ::= ([A-Z a-z] ou [0-9])+ VerNum ::= versão do protocolo (ex. 7.0)</p>	<p>(init Side Unum PlayMode)</p> <p>Side ::= l ou r Unum ::= 1 a 11 PlayMode ::= um dos modos do jogo (error no more team or player or goalie)</p>
<p>(reconnect TeamName Unum)</p> <p>TeamName ::= ([A-Z a-z] ou [0-9])+</p>	<p>(reconnect Side PlayMode)</p> <p>Side ::= l ou r Unum ::= 1 a 11 PlayMode ::= um dos modos do jogo (error no more team or player) (error reconnect)</p>
<p>(Bye)</p>	

Tabela 5.1 – Estabelecendo e Re-estabelecendo conexão e desconectando do servidor

Se um cliente se liga sucessivamente ao servidor usando uma versão de protocolo maior ou igual a versão 7.0, o servidor, adicionalmente, envia as seguintes mensagens : uma mensagem que contém os parâmetros do servidor, uma mensagem contendo os parâmetros dos jogadores e uma mensagem contendo os tipos de jogadores. No formato descrito abaixo. Por fim, o jogador recebe uma mensagem com informação das mudanças dos demais jogadores.

- (server param gwidth inertia moment psize pdecay prand pweight pspeed max paccel max stamina max stamina inc recover init recover dthr recover min recover dec e_ort init e_ort dthr e_ort min effort dec e_ort ithr e_ort inc kick rand team actuator noise prand factor l prand factor r kick rand factor l kick rand factor r bsize bdecay brand bweight bspeed max baccel max dprate kprate kmargin ctradius ctradius width maxp minp maxm minm maxnm minnm maxn minn visangle visdist windir winforce winang winrand kickable area catch area l catch area w catch prob goalie max moves ckmargin o_side area win no win random say cnt max SayCoachMsgSize clang win size clang de_ne win clang meta win clang advice win clang info win clang mess delay clang mess per cycle half time sim st send st recv st sb step lcm st SayMsgSize hear max hear inc hear decay cban cycle slow down factor useo_side kicko_o_side o_side kick margin audio dist dist qstep land qstep dir qstep dist qstep l dist qstep r land qstep l land qstep r dir qstep l dir qstep r CoachMode CwRMode old hear sv st start goal l start goal r fullstate l fullstate r drop time)
- (player param player types subs max pt max player speed max delta min player speed max delta max stamina inc max delta factor player decay delta min player decay delta max inertia moment delta factor dash power rate delta min dash power rate delta max player size delta factor kickable margin delta min kickable margin delta max kick rand delta factor extra stamina delta min extra stamina delta max e_ort max delta factor e_ort min delta factor)

Para cada tipo de jogador existe uma forma de mensagem :

- (player type id player speed max stamina inc max player decay inertia moment dash power rate player size kickable margin kick rand extra stamina e_ort max e_ort min)

5.4.2 Controle do Cliente

Do Cliente para o Servidor	Somente uma vez por ciclo
(catch Direction) Direction ::= minmoment _ maxmoment graus	Sim
(change view Width Quality) Width ::= narrow ou normal ou wide Quality ::= high ou low	Não
(dash Power) Power ::= minpower _ maxpower De acordo com a força.	Sim
(kick Power Direction) Power ::= minpower _ maxpower Direction ::= minmoment _ maxmoment graus	Sim
(move X Y) X ::= -52.5 a 52.5 Y ::= -34 a 34	Sim
(say Message) Message ::= String	Não
(sense body) O servidor retorna (sense body Time (view mode fhigh ou lowg fnarrow ou normal ou wideg) (stamina Stamina effort) (speed AmountOfSpeed DirectionOfSpeed) (head angle HeadAngle) (kick KickCount) (dash DashCount) (turn TurnCount) (say SayCount) (turn neck TurnNeckCount) (catch CatchCount) (move MoveCount) (change view ChangeViewCount))	Não
(score)	Não

O servidor retorna (score Time OurScore TheirScore)	
(turn Moment) Moment ::= minmoment _ maxmoment graus	Sim
(turn neck Angle) Angle ::= minneckmoment _ maxneckmoment graus Turn_neck é relativo a direção do corpo do jogador. Pode ser usado no mesmo ciclo que os comandos turn, dash ou kick.	Sim

Tabela 5.2 – Tabela com os Comando de Controle do Cliente

5.4.3 Protocolo de Sensores do Cliente

Do Servidor para o Cliente
<p>(hear Time Sender "Message") (hear Time Online Coach Coach Language Message)</p> <p>Time ::= ciclo do Simulador Soccer Server Sender ::= online coach left ou online coach right ou coach j referee ou self ou Direction Direction ::= -180 _180 graus Message ::= String Online Coach ::= online coach left ou online coach right Coach Language Message ::= pesquisar a Sintaxe da Linguagem Standard dos Treinadores On-line</p>
<p>(see Time ObjInfo+)</p> <p>Time ::= ciclo do Simulador Soccer Server ObjInfo ::= (ObjName Distance Direction DistChange DirChange BodyFacingDir HeadFacingDir) Ou (ObjName Distance Direction DistChange DirChange) Ou (ObjName Distance Direction) Ou (ObjName Direction) ObjName ::= (p ["Teamname" [UniformNumber [goalie]]) Ou (b) Ou (g [l ou r]) Ou(f c) Ou (f [l ou c ou r] [t ou b]) Ou (f p [l ou r] [t ou c ou b]) Ou (f g [l ou r] [t ou b]) Ou (f [l ou r ou t ou b] 0) Ou (f [t ou b] [l ou r] [10 ou 20 ou 30 ou 40 ou 50]) Ou (f [l ou r] [t ou b] [10 ou 20 ou 30]) Ou (l [l ou r ou t ou b]) Ou (B) Ou (F) Ou (G) Ou (P) Distance ::= número real positivo Direction ::= -180 a 180 graus DistChange ::= número real</p>

DirChange ::= número real HeadFaceDir ::= -180 a 180 graus BodyFaceDir ::= -180 a 180 graus Teamname ::= String UniformNumber ::= 1 a 11
(sense body Time (view mode fhigh ou lowg fnarrow ou normal ou wideg) (stamina Stamina effort) (speed AmountOfSpeed DirectionOfSpeed) (head angle HeadAngle) (kick KickCount) (dash DashCount) (turn TurnCount) (say SayCount) (turn neck TurnNeckCount) (catch CatchCount) (move MoveCount) (change view ChangeViewCount)) Time ::= ciclo do Simulador Soccer Server Stamina ::= número real positivo Effort ::= número real positivo AmountOfSpeed ::= número real positivo DirectionOfSpeed ::= -180 a 180 graus HeadAngle ::= -180 a 180 graus *Count ::= inteiro positivo

Tabela 5.3 – Tabela com os Protocolos de Sensor do Cliente

5.5 Sensores

Um agente RoboCup tem três sensores diferentes. O sensor auditivo que detecta mensagens enviadas pelo árbitro, treinador e outros jogadores.

O sensor visual que detecta informação visual do campo, como distância e a direção de determinados objetos no campo de visão do jogador. O sensor visual funciona também como um sensor de proximidade (detectando objetos que estão perto, mas não objetos que

estejam atrás do jogador), e o sensor corporal , que detecta o estado físico do jogador, como sua energia, velocidade e ângulo de rotação do pescoço. Juntos, estes sensores transmitem ao jogador uma noção razoavelmente boa do ambiente.

5.5.1 Sensor Auditivo

As mensagens de áudio são enviadas cada vez que um cliente (jogador) ou treinador, envia um comando “say” ou “hear” por exemplo.

O formato das mensagens de áudio enviadas pelo servidor são do tipo :

(hear Time Sender “Message”)

“Time” - ciclo atual do Simulador Soccer Server.

“Sender” - identifica o remetente da mensagem .

Tendo os seguintes tipos de identificações :

- self – identifica um jogador como remetente da mensagem.
- referee – identifica o árbitro como remetente da mensagem.
- Online coach left ou online coach right – Caso seja uma mensagem enviada por um dos treinadores on-line.

A “Message” é a mensagem a ser enviada em questão . O tamanho máximo da mensagem está definido na variável say_msg_size em bytes na tabela de parâmetros abaixo.

Parâmetros no server.conf	Valores
Audio_cut_dist	50.0
hear max	2
hear inc	1
hear decay	2
Say msg size	512

Tabela 5.4 : Parâmetros do sensor auditivo

Existem uma série de limitações quanto ao sensor auditivo dependendo da configuração dos parâmetros. Um mensagem transmitida por um jogador, é transmitida apenas para os jogadores que estiverem no raio de `audio_cust_dist` que é dado em metros. Em cada ciclo a capacidade auditiva é incrementada `hear_inc` até chegar ao máximo de `hear_max`. Para evitar que uma equipe controle o canal de comunicação as equipes tem diferentes capacidades auditivas. Com as configurações da tabela, um jogador pode ouvir no máximo duas mensagens a cada ciclo de simulação .

5.5.2 Sensor Visual

O sensor visual transmite ao jogador informações sobre o que ele pode ver . Esta informação é enviada automaticamente para o jogador a cada `sense_step` (a cada 150 milisegundos).

A informação visual é transmitida da seguinte forma :

(see **ObjName Distance Direction DistChng DirChng BodyDir HeadDir**)

Cada robô tem um determinado campo de visão que depende de uma série de fatores. Em primeiro lugar estão os parâmetros do servidor o `sense_step` e o `visible_angle`, que especificam o intervalo de tempo entre cada informação visual, e o ângulo de visão do robô, respectivamente.

O jogador pode influenciar a frequência e a qualidade da informação alterando o `ViewWidth` e o `ViewQuality`, respectivamente.

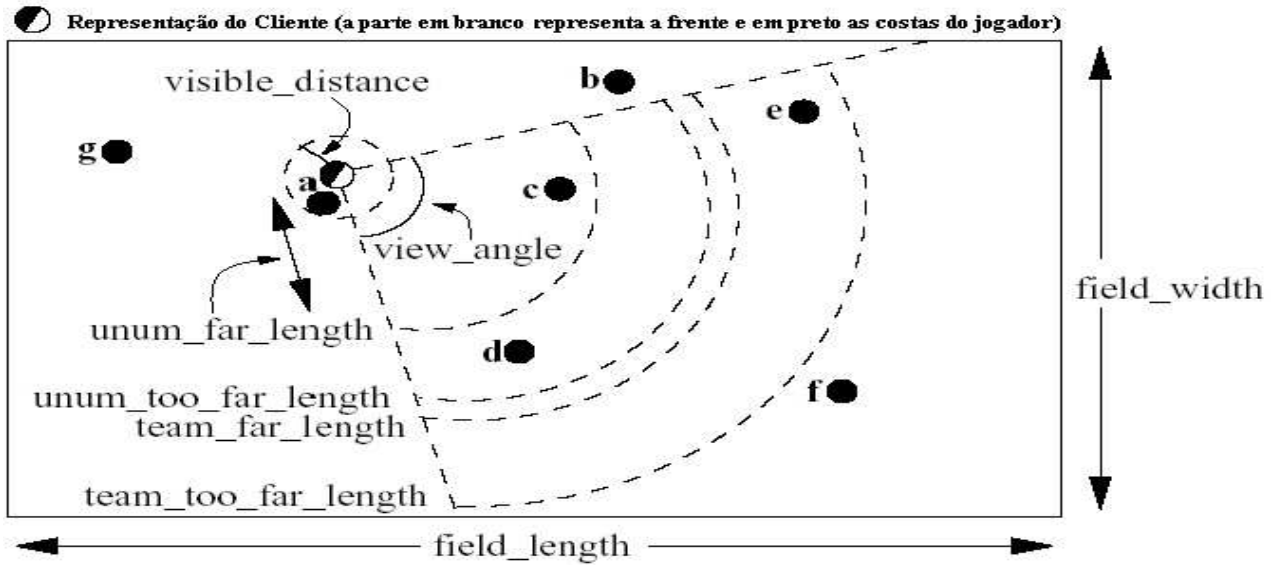


Figura 5.2 – Representação das variáveis responsáveis pela visão de um jogador

O agente jogador na figura 5.2, está representado em dois semi-círculos. A parte mais clara é a frente do jogador. Os círculos em preto representam os objetos em campo. O agente só consegue ver os objetos dentro do seu ângulo de visão (`view_angle`) e os que estão a uma distância visível (`visible_distance`), note que objetos que se encontram nas costas do jogador não podem ser vistos por ele.

As variáveis da figura acima como : `unum_far_length` , `unum_too_far_length`, `team_far_length` e `team_too_far_length` , dizem respeito a capacidade visual do jogador de identificar ou não o número do jogador e seu time a uma dada distância.

Parâmetros no server.conf	Valor
<code>Sense_step</code>	150
<code>Visible_angle</code>	90.0
<code>Visible_distance</code>	3.0
<code>Unum_far_length</code>	20.0
<code>Unum_too_far_length</code>	40.0
<code>Team_far_length</code>	40.0
<code>Team_too_far_length</code>	60.0
<code>Quantize_step</code>	0.1
<code>Quantize_step_l</code>	00.1

Tabela 5.5 – Parâmetros do sensor visual

5.4.3 Sensor Corporal

Os sensores corporais transmitem ao agente o seu estado “físico”. Essa informação é automaticamente enviada para o jogador segundo `sense_body_step`, normalmente a cada 100 milissegundos.

O formato da mensagem encontra-se abaixo :

(**sense body Time**
 (**view mode ViewQuality ViewWidth**)
 (**stamina Stamina E_ort**)
 (**speed AmountOfSpeed DirectionOfSpeed**)
 (**head angle HeadDirection**)
 (**kick KickCount**)
 (**dash DashCount**)
 (**turn TurnCount**)
 (**say SayCount**)
 (**turn neck TurnNeckCount**)
 (**catch CatchCount**)
 (**move MoveCount**)
 (**change view ChangeViewCount**))

Os valores ou significados das variáveis encontram-se na tabela abaixo :

Variáveis	Valor ou Significado
ViewQuality	high ou low
ViewWidth	narrow, normal ou wide
AmountOfSpeed	Valor aproximado da velocidade do jogador
DirectionOfSpeed	Valor aproximado da direção do jogador
HeadDirection	Valor referente a direção da cabeça do jogador.

Tabela 5.6 – Parâmetros do sensor corporal

As variáveis Count são o número de comandos, de cada tipo, executados pelo servidor. Por exemplo , “DashCount = 125” significa que o jogador já executou 125 comandos dash.

5.6 Modelos de Ações

5.6.1 Agarrar (“Catch”)

O goleiro é o único jogador com a habilidade de agarrar a bola. O goleiro pode agarrar a bola no jogo; se a bola estiver dentro de sua área de atuação e o goleiro dentro da área de sua respectiva defesa. A área de defesa é uma área retangular de comprimento l (catchhable_area_l) e largura w (catchhable_area_w). (como mostra a Figura abaixo). A bola será pega com base ainda em uma probabilidade (catchhable_probability) se estiver dentro desta área (jamais será pega se estiver fora dela).

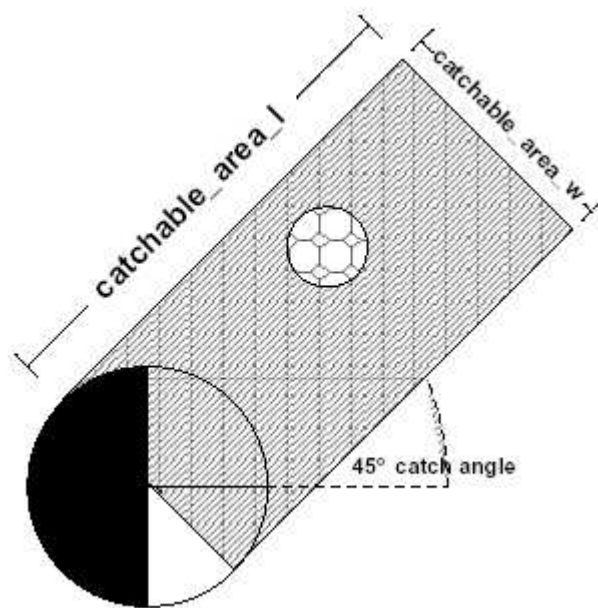


Figura 5.3 – Representação da área de atuação de um goleiro

Se um comando de defesa for mal sucedido, faz-se catch_ban e, o ciclo se tornará proibitivo para novas defesas até que um outro comando seja usado (os comandos de defesa

durante este tempo não têm efeito nenhum). Se o goleiro defender a bola, a modalidade do jogo atualizará as variáveis `goalie catch ball [l|l|]' e em seguida a `free kick [l|l|]', durante o mesmo ciclo. Uma vez que o goleiro defendeu a bola, pode-se usar o comando move para mover-se com a bola dentro da área de defesa do campo. O goleiro pode usar o comando move até um número finito de vez determinado pela variável goalie_max_moves antes de recolocar a bola em jogo, com um chute por exemplo.

Note que defender a bola , mover-se, largar a bola a uma distância curta e defendê-la novamente imediatamente procurando como se diz matar tempo é identificado no jogo como trapaça ou melhor dizendo obstrução.

Parâmetros no server.conf	Valor
Catchable_area_l	2.0
Catchable_area_w	1.0
Catch_probability	1.0
Catch_ban_cycle	5
Goalie_max_moves	2

Tabela 5.7 – Parâmetros da área de atuação do goleiro

5.6.2 Aceleração (“Dash”)

O comando de aceleração é usado para acelerar o jogador no sentido de suas tomadas de aceleração do corpo, utiliza o poder (power) como um parâmetro de aceleração. A escala válida para o power da aceleração pode ser configurado em server.conf, os parâmetros respectivos são minpower e maxpower. Para os valores atuais dos parâmetros para o modelo de colisão, veja tabela 5.9 .

Cada jogador tem uma determinada quantidade de força a ser consumida por comando executado. No começo, a força de um jogador é ajustada sendo o atributo stamina_max. Se um jogador acelerar para a frente (power > 0), a força será segundo sua intensidade (power). Aceleração para trás (power < 0) é mais caro para o jogador: a força é reduzida -2 .

Se a força do jogador for mais baixa do que o power necessário para a ação, o power será reduzido de modo que o comando da ação não necessite mais força além do que há

disponível. Os jogadores heterogêneos usarão força extra cada vez que o power disponível é mais baixo que a força necessária. A quantidade de força extra depende do tipo do jogador , os atributos que demonstram o mínimo e o máximo de força extra são : `extra_stamina_delta_min` e `extra_stamina_delta_max`.

Depois de reduzir a força, o servidor calcula o poder de aceleração efetivo para o comando de aceleração. O poder de aceleração efetivo (`pcf`) depende `dash_power_rate` e do comando `dash` . O esforço de um jogador é um valor entre `effort_min` e `effort_max`; é dependente da força do jogador ($pcf = effort \times dash_power_rate \times power$).

5.6.2.1 Modelo de força ou energia (“Stamina”)

Para a força de um jogador, há três variáveis importantes: o valor de força, o fator recuperação e o esforço. A força é diminuída ao colidir e é preenchida ligeiramente a cada ciclo. A recuperação é responsável por quanto a força se recupera a cada ciclo, e o esforço diz quanto efetivo será a aceleração. Parâmetros importantes para o modelo de força é são encontrados nos arquivos `server.conf` e `player.conf`, do Soccer Server. O fator recuperação só recupera 1.0 ponto a cada paralização do jogo, mas não será aumentado no transcorrer da partida.

5.6.3 Chutar (“Kick”)

O comando de chute leva apenas dois parâmetros, o poder do chute (`power`) que varia entre `minpower` e `maxpower` e o ângulo do chute do jogador em relação a bola.

O ângulo é determinado em graus e deve estar entre `minmoment` e `maxmoment` (veja a tabela de parâmetro atuais).

Uma vez que o comando `kick` chegue ao servidor, o chute será executado se a bola for capaz de ser chutada pelo jogador e o jogador não estiver à direita marcado. A bola é chutada se o jogador for capaz, se a distância entre o jogador e a bola estiver entre 0 e `kick_able_margin`. Jogadores heterogêneos podem ter margens de chute diferentes.

5.6.4 Movimentar-se (“Move”)

O comando `move` pode ser usado para colocar um jogador diretamente sobre uma posição desejada em campo. `move` existe para montar o time antes do início do jogo, não funciona durante jogo. Torna-se disponível quando a bola sai do campo (modo de jogo `before_kick_off`) e depois que um gol for marcado (modos de jogo `goal_r_n` ou `goal_l_n`). Nestas situações, os jogadores podem ser colocado em qualquer posição dentro do campo podendo ser movidos quantas vezes forem necessárias, contanto que o modo de jogo não mude. Um segundo propósito do comando de `move` é mover o goleiro para a cobrança de penalidade para que o goleiro possa pegar a bola. Se o goleiro pegar a bola, isto possibilita a ele mover-se junto com a bola dentro da área. O goleiro é permitido mover-se `goalie_max_moves` de tempo antes de repor a bola com um chute novamente em jogo.

Parâmetros Básicos do server.conf		Parâmetros dos Jogadores Heterogêneos		
Name	Value	Name	Value	Range
<code>.minpower</code>	-100			
<code>.maxpower</code>	100			
<code>.stamina_max</code>	4000			
<code>.stamina_inc_max</code>	45	<code>.stamina_inc_max_delta_factor</code>	-100	
		<code>.player_speed_max_delta_min</code>	0.0	25
		<code>.player_speed_max_delta_max</code>	0.2	45
<code>.extra_stamina</code>	0.0	<code>.extra_stamina_delta_min</code>	0.0	0.0
		<code>.extra_stamina_delta_max</code>	100.0	100.0
<code>.dash_power_rate</code>	0.006	<code>.dash_power_rate_delta_min</code>	0.0	0.006
		<code>.dash_power_rate_delta_max</code>	0.002	0.008
<code>.effort_min</code>	0.6	<code>.effort_min_delta_factor</code>	-0.002	
		<code>.extra_stamina_delta_min</code>	0.0	0.4
		<code>.extra_stamina_delta_max</code>	100.0	0.6
<code>.effort_max</code>	1.0	<code>.effort_max_delta_factor</code>	-0.002	
		<code>.extra_stamina_delta_min</code>	0.0	0.8
		<code>.extra_stamina_delta_max</code>	100.0	1.0
<code>.effort_dee_thr</code>	0.3			
<code>.effort_dee</code>	0.005			
<code>.effort_inc_thr</code>	0.6			
<code>.effort_inc</code>	0.01			
<code>.recover_dee_thr</code>	0.3			
<code>.recover_dee</code>	0.002			
<code>.recover_min</code>	0.5			
<code>.player_accel_max</code>	1.0			
<code>.player_speed_max</code>	1.0	<code>.player_speed_max_delta_min</code>	0.0	1.0
		<code>.player_speed_max_delta_max</code>	0.2	1.2
<code>.player_rand</code>	0.1			
<code>.wind_force</code>	0.0			
<code>.wind_dir</code>	0.0			
<code>.wind_rand</code>	0.0			
<code>.player_decay</code>	0.4	<code>.player_decay_delta_min</code>	0.0	0.4
		<code>.player_decay_delta_max</code>	0.2	0.6

Tabela 5.8 – Parâmetros básicos para ações de Colisão (“Dash”)

5.6.5 Dizer (“Say”)

Usando o comando say, os jogadores podem reenviar mensagens a outros jogadores. Mensagens com longos caracteres de tamanho say_msg_size (bytes), onde caráter válidos para comando são [-0-9a-zA-Z () .+ * /? <>]. As Mensagens (say) dos jogadores podem ser ouvidas dentro de uma distância de audio_cut_dist para os jogadores da mesma equipe . Mensagens (say) serão mandadas de volta e enviadas ao servidor a todos os jogadores dentro dessa distância. O uso do comando say só é restringido pela limitação dos jogadores de ouvir mensagens de longas distâncias e de tamanhos muito grandes.

Parâmetros Básicos Server.conf		Parâmetros dos Jogadores Heterogêneos		
Name	Value	Name	Value	Range
.minpower	-100			
.maxpower	100			
.min_moment	-180			
.max_moment	180			
.kickable_margin	0.7	.kickable_margin_delta_min	0.0	0.7
		.kickable-margin_delta_max	0.2	0.9
.kick_power_rate	0.027			
.kick_rand	0.0	.kick_rand_delta_factor	0.5	
		.kickable_margin_delta_min	0.0	0.0
		.kickable_margin_delta_max	0.2	0.1
.ball_size	0.085			
.ball_decay	0.94			
.ball_rand	0.05			
.ball_speed_max	2.7			
.ball_accel_max	2.7			
.wind_force	0.0			
.wind_dir	0.0			
.wind_rand	0.0			

Tabela 5.9 - Parâmetros básicos para a ação como chutar (“Kick”) e a bola

Parâmetros Básicos no server.conf	Value
.say_msg_size	512
.audio_cut_dist	50
.hear_max	2
.hear_inc	1
.hear_decay	2

Tabela 5.10 - Parâmetros básicos para ações como dizer (“say”)

5.6.6 Girar (“Turn”)

Enquanto a colisão (dash) é usada para acelerar o jogador em uma dada direção, o comando turn, é usado para mudar a direção do corpo do jogador. O argumento para o comando turn é o momento; valores válidos para o momento estão entre minmoment e maxmoment.

5.6.7 Girar o pescoço (“TurnNeck”)

Com turn_neck, um jogador pode virar seu pescoço um pouco independentemente de seu corpo. O ângulo da cabeça do jogador é o ângulo de sua visão. O comando turn muda o ângulo do corpo do jogador enquanto o turn_neck muda o ângulo do pescoço do jogador relativo a seu corpo. O mínimo e máximo ângulo relativo para o o pescoço do jogador é determinado por minmoment e maxmoment em server.conf.

Também, podem ser executados comandos turn_neck durante o mesmo ciclo que um turn, dash, ou kick. Os argumentos para turn_neck não fazem nenhum efeito para o turn. O argumento para um comando turn_neck deve estar entre minneckmoment e maxneckmoment.

Parâmetros Básicos Server.conf		Parâmetros dos Jogadores Heterogêneos		
Name	Value	Name	Value	Range
.min_moment	-180			
.max_moment	180			
.inertia_moment	5.0	.player_decay_delta_min	0.0	
		.player_decay_delta_max	0.2	5.0
		.inertia_moment_delta_factor	25.0	10.0

Tabela 5.11 - Parâmetros básicos para ação de girar (“Turn”)

Parâmetros Básicos no server.conf	Value
.minneckang	-90
.maxneckang	90
.minneckmoment	-180
.maxneckmoment	180

Tabela 5.12 - Parâmetros básicos para ação de girar o pescoço (“TurnNeck”)

5.7 Jogadores Heterogêneos

A partir do Soccer Server 7, jogadores heterogêneos foram introduzidos, onde agora o Soccer Server sorteia diversos tipos de jogadores antes de iniciar uma partida. (player_types). Estes diversos tipos de jogadores possuem habilidades diferentes baseados no arquivo player.conf. Ambas as equipes possuem diversos tipos de jogadores. O padrão para player_types é 0 (zero). Quando os jogadores conectam ao servidor, o servidor recebe a informação sobre o tipo do jogador. O treinador on-line pode mudar o jogador em modos before_kick_off e o número de substituições é determinado por subs_max durante o modo play_on a mudança no jogo acontece usando o comando change_player_type. Cada jogador substituído por outro, possui sua força, recuperação e esforço iniciados ao máximo antes do jogador entrar em campo.

5.8 Parâmetros do Soccer Server

Name	Valor Padrão (Default)	Valor Configurado no server.conf
.goal_width	7.32	14.02
.player_size		0.3
.player_decay		0.4
.player.rand		0.1
.player_weight		60.0
.player_speed_max		1.0
.player_accel_max		1.0
.stamina_max		4000.0
.stamina_inc_max		45.0
.recover_dee_thr		0.3
.recover_min		0.5
.recover_dee		0.002
.effort_dee_thr		0.3
.effort_min		0.6
.effort_dee		0.005
.effort_inc_thr		0.6

.effort_inc		0.01
.kick_rand		0.0
.team_actuator_noise		
.prand_factor_l		
.prand_factor_r		
.kick_rand_factor_l		
.kick_rand_factor_r		
.ball_size		0.085
.ball_decay		0.94
.ball_rand		0.05
.ball_weight		0.2
.ball_speed_max		2.7
.ball_accel_max		2.7
.dash_power_rate		0.006
.kick_power_rate		0.027
.kickable_margin		0.7
.control_radins		
.catch_probability		1.0
.catchable_area_l		2.0
.catchable_area_w		1.0
.goalie_max_moves		2
.maxpower		100
.minpower		-100
.maxmoment		180
.minmoment		-180
.maxneckmoment		180
.minneckmoment		-180
.maxneckang		90.0
.minneckang		-90
.visible_angle		90.0
.visible_distance		
.audio_cut_dist		50.0
.quantize_step		0.1
.quantize_step_l		
.quantize_step_dir		0.01
.quantize_step_dist_team_l		
.quantize_step_dist_team_r		
.quantize_step_dist_l_team_l		
.quantize_step_dist_l_team_r		
.quantize_step_dist_r_team_l		
.quantize_step_dist_r_team_r		
.ckick_margin		1.0
.wind_dir		0.0
.wind_force		0.0
.wind_rand		0.0
.wind_none		
.wind_random		
.inertia_moment		5.0
.half_time		300
.drop_ball_time		200
.port		6000
.coach_port		6001
.olcoach_port		
.say_coach_cnt_max		128

.say_coach_msg_size		128
.simulator_step		100
.send_step		150
..rcv_step		10
.sense_body_step		100
.say_msg_size		512
.clang_win_size		300
.clang_define_win		1
.clang_meta_win		1
.clang_advice_win		1
.clang_info_win		1
.clang_mess_delay		50
.clang_mess_per_cycle		1
.hear_max		2
.hear_inc		1
.hear_decay		2
.catch_ban_cycle		5
.coach		5
.coach_w_referee		
.old_coach_hear		
.send_vi_step		100
.use_offside		On
.offside_active_area_size		5
.forbid_kick_off_offside		On
.log_file		
.record		
.record_version		3
.record_log		On
.record_messages		
.send_log		On
.log_times		Off
.verbose		Off
.replay		
.offside_kick_margin		9.15
.slow_down_factor		
.start_goal_l		
.start_goal_r		
.fullstate_l		
.fullstate_r		

Tabela 5.13 - Parâmetros básicos do Simlador Soccer Server

6. SOCCER MONITOR

6.1 Fazendo Download

O Soccer Monitor possui diversas versões, espalhadas pela rede, mas algumas versões encontra-se disponível para download no site <http://sserver.sourceforge.net> .

Clique sobre o tópico Downloads localizado na frame do lado esquerdo, automaticamente a frame do lado direito carregará a seção de downloads onde lá você encontrará THE ROBOCUP SOCCER SIMULATOR MONITOR, clique sobre o link para fazer seu download.

O Link clicado o levará a uma nova página com diversas versões do Soccer Monitor, escolha uma e inicie seu download.

6.2 Apresentação

O Soccer Monitor é a parte do simulador que provê a interface visual da partida. Usando o monitor podemos vivenciar o jogo e controlar procedimentos. Existem vários tipos de visualizadores com características diferentes e que rodam em diversas plataformas. Tipicamente um visualizador fornece as seguintes informações :

- O resultado no decorrer do jogo;
- O nome das equipes;
- A posição da Bola;
- A possibilidade de gravar um jogo.

De forma colaborativa o LogPlayer (ferramenta comentada no tópico 5.4), e o Monitor permitem que se possa rever jogos (semelhante a repetição de jogos na televisão), muito útil em momentos de investigação, análise e debug de clientes.

Visualizadores mais sofisticados, e já com o objetivo de facilitar o desenvolvimento de equipes mais robustas, permitem, por exemplo que se visualizem os ângulos de visão dos robôs e as mensagens que eles transmitem. Também tem sido desenvolvidos projetos na construção de visualizadores 3D.

A comunicação entre o Monitor e o Servidor é feita através da comunicação do tipo UDP/IP porta 6000 (default). O servidor envia informações ao visualizador em cada ciclo (um ciclo tem um período de 100 milissegundos). É possível ligar vários visualizadores ao mesmo servidor permitindo assim que o mesmo jogo seja transmitido em vários terminais .

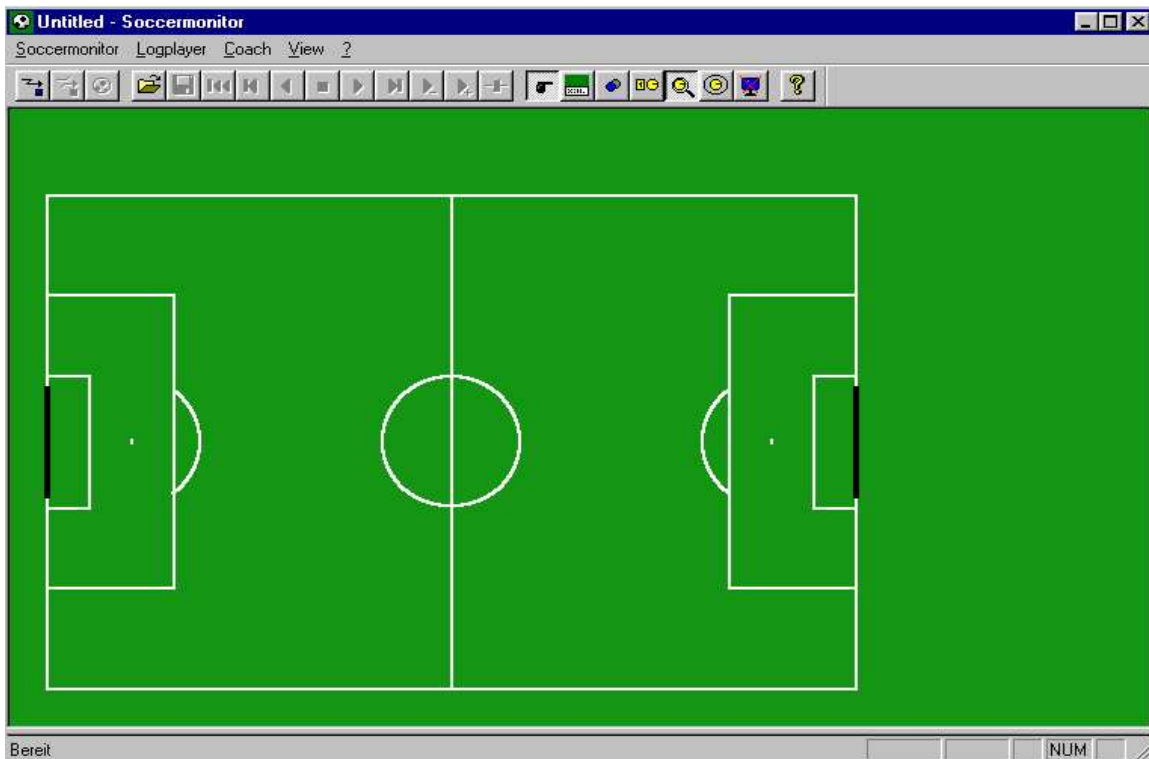


Figura 6.1 : Interface de um Modelo de Soccer Monitor

6.3 Sistema de Coordenadas

É o mesmo sistema de coordenadas cartesianas tradicional segundo a convecção vertical é positivo para cima e horizontal é positivo para a direita. Cada time possui seu próprio sistema de coordenadas, fazendo com que, na visão do telespectador, os sistemas de coordenadas dos dois times sejam simétricos em relação ao centro.

6.4 Tempo, Contagem de Ciclos e Troca de Dados

No futebol simulado, o tempo não corre como na realidade. As unidades de tempo são os ciclos de processamento do servidor. Cada ciclo dura aproximadamente um décimo de segundo. Neste tempo, o servidor processa as chamadas-texto recebidas de cada jogador, converte-as na situação física atual do jogo, envia os dados das novas posições de cada jogador e da bola para os monitores, e envia os dados específicos relativos a cada jogador. Assim, cada jogador recebe somente o “seus” dados de visão, audição e velocidade.

Tal como num jogo de futebol real, o jogo simulado é dividido em dois tempos de três mil ciclos cada. O que dá uma duração de aproximadamente cinco minutos para cada tempo.

Informações pertinentes ao andamento do jogo, tais como saídas de bola, escanteios, início e fim de partida são avisadas pelo servidor a todos os clientes.

6.5 As Bandeiras de Indicação no Campo (Flags)

Jogadores reais se localizam dentro do campo de futebol utilizando sua visão, e a memória sobre suas últimas posições visitadas. Da mesma maneira, os jogadores autônomos também se localizam no campo digital do ambiente se utilizando de referências. Estas referências são as bandeiras e linhas de localização (flags).

O Soccer Monitor para ser iniciado espera que no mínimo já tenha sido iniciado o Soccer Server . Cumprida essa etapa, você precisa conectar o Monitor ao Servidor , o que pode ser feito clicando no primeiro ícone apresentado no canto esquerdo da tela ou acessando o menu Soccermonitor e em seguida clicando em Connect to Server.

Você deverá ver uma tela como esta :

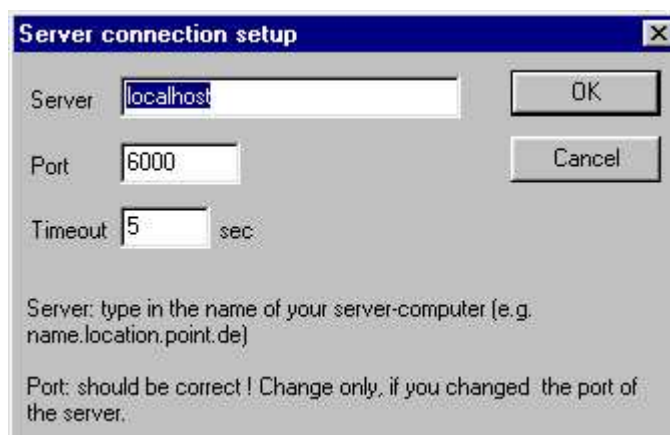


Figura 6.3 : Janela de conexão entre o Soccer Monitor e o Soccer Server

O parâmetro Server solicita o IP da máquina Servidora, como no nosso caso estamos efetuando uma simulação em uma mesma máquina usamos o padrão localhost ou 127.0.0.1, já quanto a porta de comunicação usaremos a definida pela RoboCup a porta de número 6000. O parâmetro TimeOut é o tempo de espera pelo Monitor em relação a obtenção da conexão com o servidor, seu valor varia entre 2 a 30 segundos de espera.

Então mantenha os parâmetros, como estão na figura e clique em OK .

Feito isso o Monitor estará conectado ao Servidor, e este responsável pelas mudanças que ocorreram daqui para frente.

Caso não haja nenhum Cliente, isto é, nenhum agente jogador, sido executado o servidor junto com o monitor entram em estado de espera, aguardando a chega dos jogadores . Sugiro que no caso de “marinheiros de primeira viagem” , que desejem iniciar uma partida mesmo sem antes ter criado uma equipe ou jogador, que vá até a biblioteca de agentes (discutida no tópico 7.2 deste trabalho) e faça o download de um simples agente jogador.

A partir do momento que conecta-se um jogador ao servidor Soccer Server, o mesmo envia os dados de atualização do ambiente inserindo esse jogador em campo, mostrando no alto da tela o placar com o nome da equipe em campo.

Tendo sido todos os jogadores executados e em campo, para iniciar a partida clique no terceiro ícone da esquerda para a direita (ícone de uma Bola) ou entre no menu Soccermonitor e em seguida clique em Kick-off, pronto você iniciou uma partida, tenha um bom jogo.

7. SOCCER CLIENTE

7.1 Apresentação

O Soccer Cliente corresponde a área onde é implementado o agente jogador de futebol. Para se criar agentes jogadores nesta área principalmente para iniciantes no assunto partindo-se do zero é uma tarefa não só desafiadora como uma “tolice”, haja vista que existe uma biblioteca de agentes simples prontos disponível .

Após entrar na biblioteca de agentes, e baixar este agente escolhido, você pode até executá-lo , não esquecendo de primeiro executar o Soccer Server e em seguida o Soccer Monitor, para enfim executar o seu Soccer Cliente.

No Soccer Cliente, o desenvolvedor define o comportamento e as ações do agente jogador. Comportamentos e ações estas que vão deste movimentar-se pelo campo , chutar ou passar a bola até a criação de estratégias que definam funções a cada jogador, como por exemplo, a de um atacante ou zagueiro em campo.

Criar agentes jogadores não é uma tarefa nada fácil, exige do desenvolvedor conhecimentos sobre diversos assuntos, como por exemplo arquitetura de agentes, tema abordado nos tópicos seguintes , conhecer o que já foi feito e como o fazem , os comandos existentes, as limitações por se estar desenvolvendo algo em tempo-real, são passos fundamentais antes de se aventurar na implementação de uma equipe de futebol de robôs simulados.

7.2 Bibliotecas de agentes

Para auxiliar o desenvolvimento, e não perder tempo com problemas de comunicação encontra-se disponível no site <http://sserver.sourceforge.net> , bibliotecas de clientes de diversas versões do Soccer Server , desenvolvidas em diversas linguagens.

No tópico 8.5, é apresentado de forma rápida a arquitetura da biblioteca Krislet, retirada da biblioteca de agentes, desenvolvida em Java , com o objetivo de acelerar o primeiro contato para desenvolvedores iniciantes na área, antes que os mesmos metam a “mão na massa”.

7.3 Características de um agente jogador

Um agente jogador é dotado de algumas características tais como :

Características	Significado
Autônomo	Controla suas próprias ações
Pró-ativo (Orientado a Objetivos)	Segue objetivos e não apenas reage a mudanças no ambiente
Pode ser flexível ou não	Em relação ao conjunto de ações pré-definidas
Possui ou não a capacidade de aprender	Mudança no comportamento baseado em experiências anteriores
É Comunicativo	Comunica-se com outros agentes
Temporalmente Contínuo	É um processo executado continuamente

Tabela 7.1 – Tabela com as características de um agente jogador

Embora muitos pesquisadores na área desenvolvam agentes jogadores de futebol, rotular um agente como inteligente exige que tal agente possua a característica de flexibilidade e capacidade de aprender, onde poucas equipes desenvolvidas até hoje chegaram.

Em sua grande maioria as equipes desenvolvidas, possuem um conjunto simples fixo de regras pré-definidas e deixam a desejar em relação a capacidade de aprender . A preocupação maior está em posicionar a equipe em campo de modo que joguem da maneira mais próxima de uma partida de futebol entre humanos, isto é, obedecendo posicionamentos , funções em campo como agarrar, defender e atacar.

7.4 Arquitetura Clássica de Agente

A arquitetura de agentes é definida por [Wooldridge, Jennings, 1994] como : “uma metodologia particular para definir agentes. Especifica como o agente pode ser decomposto na construção de um ambiente em módulos e como estes módulos podem interagir. O conjunto de módulos e suas interações devem prover uma resposta para a questão de como os sensores e o estado interno corrente do agente determinam suas ações e o futuro estado. Uma arquitetura deve prever as técnicas e algoritmos para suportar esta metodologia. ”

De acordo com [Dhein, 2000], as arquiteturas de agentes são a ligação entre as especificações teóricas e a obtenção de resultados práticos, na medida em que buscam a implementação de sistemas segundo tais especificações.

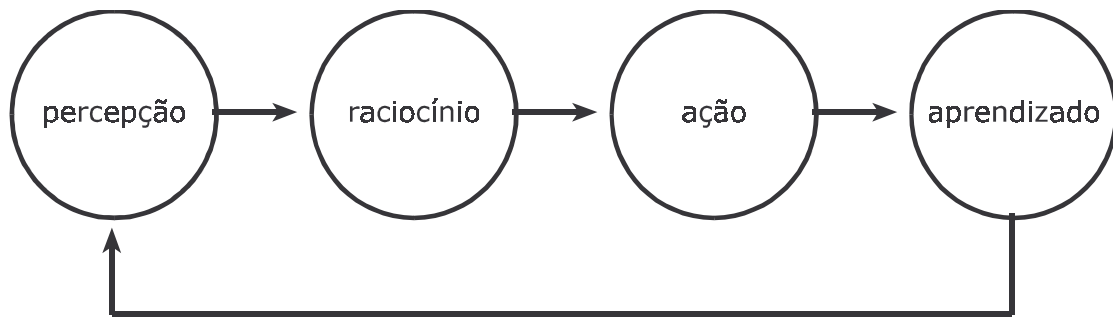


Figura 7.1 – Modelo da Arquitetura Clássica de Agentes de Russ95

7.4.1 Percepção

Para determinar qual ação tomar, um agente deve primeiro saber em que estado se encontra, para isso o agente consulta seus sensores . Os *sensores*, são as unidades que extraem do ambiente dados relevantes ao agente e os quantificam de uma maneira conveniente.

A mensagem recebida é analisada gramaticalmente através do seu “parser”. O parser tem por característica analisar a mensagem recebida, classificando os parâmetros dessa mensagem. A partir da mensagem é possível reconhecer sua natureza (visão, audição e estado do corpo) e identificar informações como : tipo do objeto, distância e direção. A figura abaixo apresenta parte de uma mensagem visual (see) recebida do servidor e a decomposição da mesma após o parser.

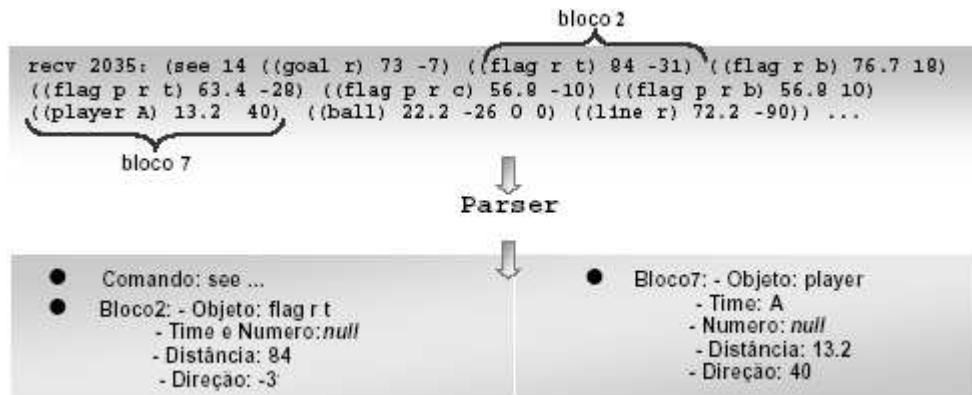


Figura 7.2 – Figura ilustrativa de como as mensagens são interpretadas

As informações presentes na mensagem informam ao agente cada objeto visível por ele. O servidor tendo enviado a informação do ambiente a todos os agentes o estado do jogo é atualizado.

No caso do simulador Soccer Server, a informação visual é relativa apenas ao campo de visão do cliente, mas é recebida sob a forma de strings (na verdade, S-expressões Lisp) periodicamente, com uma frequência que é limitada por parâmetros da qualidade da informação que se deseja (no cliente) e configuração (no servidor), mas que normalmente variam entre 37.5 milissegundos e 300 milissegundos.

Também as informações sobre a propriocepção, isto é, o estado do cliente, são recebidas, mas estas últimas apenas quando o cliente indicar que assim o deseja, e não regularmente como as visuais.

7.4.1.1 Memória

O agente jogador fazendo uso da sua percepção pode criar um modelo representativo do ambiente através de uma memória. Essa memória permite ao jogador relembrar o estado

do jogo nos últimos ciclos. A representação do ambiente armazenado na memória fica parecida com a figura a seguir.

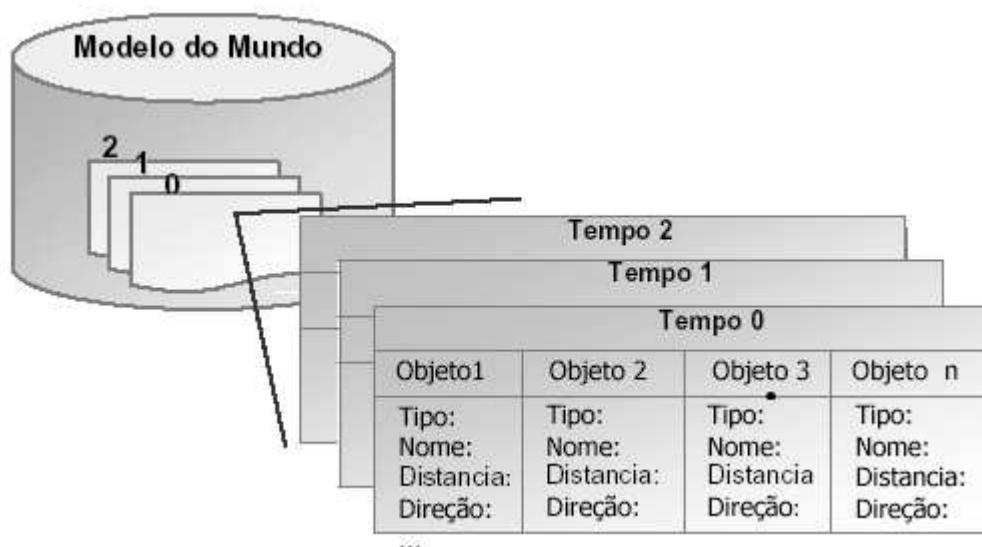


Figura 7.3 – Representação do ambiente sendo armazenado em memória pelo jogador

Essa informação é usada quando necessária, como por exemplo, se a bola não estiver visível no ciclo atual, a última informação conhecida da bola pode ser usada, ou ainda, feitas combinações entre as informações durante os três últimos ciclos que possam verificar se um jogador (companheiro ou adversário) está se aproximando da bola.

7.4.2 Raciocínio

Durante a fase de raciocínio, os dados adquiridos dos sensores serão examinados e processados, baseados no conjunto de regras, informação anterior recuperada da memória ou qualquer outro método que o agente disponha afim de determinar qual o próximo passo a ser tomado. É justamente nesse ponto onde os agentes implementam sua “inteligência”, desde os julgamentos mais primitivos até as abstrações mais complexas. Os agentes que apresentam essa camada são ditos *cognitivos*, uma vez que chegam a uma conclusão sobre o ambiente, e conseqüentemente, sobre si próprios, baseados na análise dos dados disponíveis. Em contraste temos os agentes *reativos*, que simplesmente disparam respostas a estímulos do ambiente. Por exemplo, um robô que joga na defesa e que vê a bola em sua frente poderia

simplesmente chutá-la, tentando evitar um gol do adversário. Esses agentes tendem a responder muito mais rapidamente a certas situações críticas, mas não podem planejar ações, isto é, conceber conjuntos de passos para desenvolver uma determinada intenção, o que pode limitar seriamente sua performance enquanto jogadores de futebol, em particular num ambiente onde a consciência dos outros agentes – principalmente a dos companheiros – é essencial para um bom desempenho.

A camada de raciocínio não é implementada em nenhum nível no servidor Soccer Server, uma vez que cada cliente deve absorver ou descartar as informações que achar necessárias, e tomar as medidas que considerar cabíveis, baseados em seu próprio conjunto de regras.

7.4.3 Ação

Após a análise das condições do agente e do seu ambiente, e baseado no seu conjunto de regras, é finalmente tomada a decisão sobre que ação deverá ser efetuada a seguir. Essa ação pode sempre incluir a possibilidade do agente continuar no estado atual, isto é, de não fazer nada. O agente atua no ambiente através dos efetadores, que exercem um papel inverso dos sensores: transformar intenções do agente em ações efetivas sobre os elementos do ambiente.

Os efetadores no simulador Soccer Server são strings ou S-expressões, que indicam o desejo do agente de realizar alguma ação no ambiente de jogo. Essas strings podem indicar que o agente deseja correr, chutar a bola, girar o corpo ou o pescoço, falar algo ou simplesmente requisitar informações sobre seu estado. Esses comandos recebem parâmetros que devem ser especificados pelos agentes, e podem falhar devido a restrições impostas sobre o ambiente pelo simulador, assim como certas ações não podem ser feitas no mundo físico, como chutar uma bola que se encontra a 10 metros de distância do agente. Cabe aos agentes determinar se é possível ou não realizar as ações desejadas na fase de raciocínio, uma vez que o servidor simplesmente ignora comandos que não possam ser efetutados.

7.4.4 Aprendizado

Alguns agentes ajustam seu conjunto de regras baseados nos resultados de suas ações sobre o ambiente. Esses agentes monitoram o ambiente antes e depois da execução de cada uma de suas ações para determinar se agiram de uma maneira adequada, e reforçam ou penalizam certos conjuntos de regras baseados nesse julgamento. Tais agentes são ditos agentes com aprendizado on-line, ou feedback on-line, e normalmente tendem a não repetir erros anteriores, uma característica certamente bastante desejável.

7.5 Arquitetura Krislet

7.5.1 Apresentação

Krislet é um cliente ou melhor um agente jogador, simples, feito em linguagem Java que foi extraído da biblioteca de agentes já mencionada anteriormente.

Desenvolvido por Krzysztof Langner em 1997, este tipo de jogador não possui qualquer tipo de consciência coletiva, e basicamente segue o seguinte ciclo, mostrado na figura a seguir :

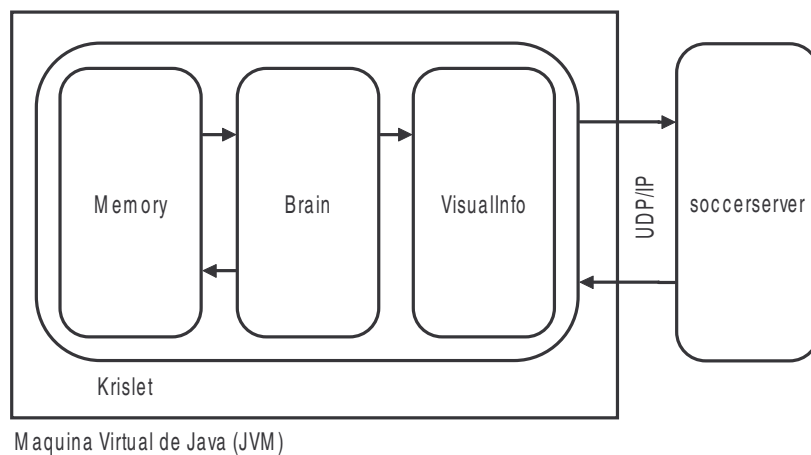


Figura 7.4 – Representação da Arquitetura do agente Krislet

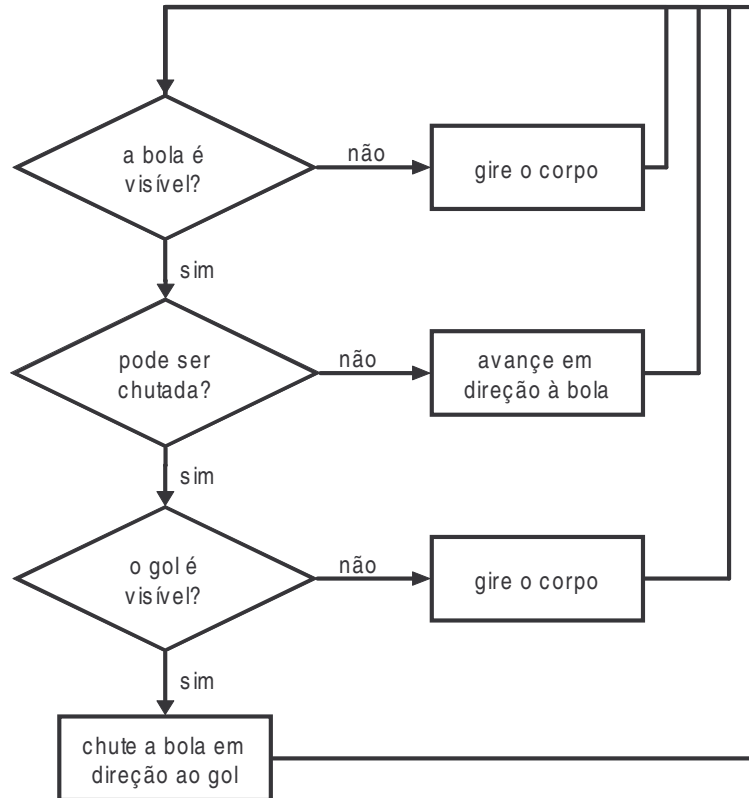


Figura 7.5 : Representação do funcionamento do jogador Krislet

7.5.1.1 Classe Krislet

A classe Krislet é responsável por estabelecer a comunicação via socket UDP/IP e além de implementar a interface SendCommand, com todos os comandos primitivos do Simulador Soccer Server tais como : move, turn, dash , kick say, changeView .

7.5.1.2 Classe Brain

A classe Brain, é a unidade de raciocínio do modelo onde todas as ações do jogador são inseridas, funciona sob uma Thread e implementa a interface SensorInput com os comandos primitivos do sensor como : see e hear .

7.5.1.3 Classe Memory

A classe Memory, possibilita ao cliente armazenar as posições dos objetos no ambiente permitindo mais tarde sua localização.

7.5.1.4 Classe VisualInfo

A classe VisualInfo é responsável pelo parsing da informação visual recebido do Soccer Server.

7.5.1.5 Outras Classes

Apesar das principais , classes já foram apresentadas existem outras classes dentro da estrutura Krislet . Como a classe ObjectInfo que estabelece informações sobre objetos como jogadores, bola, goleiras (traves) e bandeiras (flags) , levando em consideração os seguintes atributos : tipo, distância, direção, distChange, dirChange. Classes como PlayerInfo, GoalInfo, BallInfo e FlagInfo estanciam a classe ObjectInfo setando as variáveis específicas do objeto tratada pela classe.

7.5.2 Desempenho

O agente jogador Krislet possui uma arquitetura simples, regido por um conjunto de regras if then else , não se preocupa com estratégias coletivas, mas correspondente de forma rápida as mudanças no ambiente em direção ao seu objetivo simples de fazer “gols”.

Quando colocado em ambiente de jogo contra equipes que são extensões dela como é o caso da versão Sprislet, em termos de desempenho no jogo acabam vencendo devido o desempenho da equipe Sprislet ter diminuído com o aumento da complexidade quanto a percepção, raciocínio, ação e aprendizado.

7.6 Estratégia de novas Arquiteturas mais elaboradas

Depois de ter visto a simplicidade da arquitetura do agente Krislet, você acaba concluindo o quão difícil seria se todos os jogadores fossem dessa mesma forma, onde todos correriam atrás da bola, sem posicionamento em campo e esquecendo do trabalho coletivo entre os jogadores.

Lembrando que um agente jogador deve analisar de forma rápida e constante as tarefas ao qual for submetido, adaptando-se ao meio ao qual foi inserido e agindo de forma autônoma.

Partindo desse princípio, sugiro e apresento parte dos trabalhos de [Ludwig e Cordenonsi, 2002] e [Cordenonsi e Hammel, 2001] sob uma abordagem de agentes reativos, devido eficiência na realização das tarefas levando-se em conta o curto intervalo de tempo do ciclo Soccer Server.

7.6.1 Arquitetura Reativa

Segundo [Wolldridge, Jennings, 1994], uma arquitetura reativa não inclui qualquer tipo de modelo simbólico do mundo e não utiliza raciocínio simbólico complexo, somente reage as ações que ocorrem no ambiente.

7.6.2 Arquitetura de Subordinação

A arquitetura de subordinação, desenvolvida por Rodney Brooks [Brooks, 1986], surgiu da necessidade de criar um mecanismo para controlar robôs móveis e autônomos, capazes de agir em um ambiente dinâmico, onde as condições mudam rapidamente.

Assim, Brooks teve que fugir dos modelos padrões simbólicos, e passar a desenvolver um sistema baseado em comportamentos realizadores de tarefas (*task achieving behaviors*).

Nos modelos tradicionais, o comportamento dos agentes é dividido em módulos independentes, que se sucedem verticalmente em forma de cadeia, ou seja, as informações chegam do ambiente através de sensores e, após passar por todos os módulos, retornam ao

ambiente em forma de ações. O problema é dividido em pedaços, logo, cada subproblema é resolvido para depois compor a solução.

A arquitetura de subordinação, apresenta uma hierarquia de comportamentos, onde cada comportamento compete com os outros para ter controle sobre o robô. A arquitetura apresenta camadas inferiores, que representam tipos de componentes mais simples (como desviar de obstáculos), e possuem precedência sobre as camadas mais altas da hierarquia, que apresentam comportamentos mais complexos.

As camadas são construídas usando conjuntos de pequenos processadores que enviam mensagens entre si. Cada processador, ou módulo, é uma máquina de estados finitos rodando de forma assíncrona, monitorando suas entradas e enviando mensagens para suas saídas. Não há a presença de um controle central entre os processadores, onde cada módulo faz o possível para realizar sua tarefa com êxito.

Brooks comprovou que sistemas resultantes desta arquitetura, embora muito simples, executam procedimentos que seriam imprecisos se executados por sistemas simbólicos de IA [Wolldridge, Jennings, 1994].

7.6.2.1 Cooperação sem Deliberação

Após a publicação da arquitetura de subordinação [Brooks, 1985], várias pesquisas começaram a aparecer em torno de sistemas baseados em comportamentos. Assim, surgiu a arquitetura de Werger [Werger, 1998], que se baseou em Brooks para ampliar seus estudos e criar sua própria abordagem reativa.

O objetivo de Werger é trabalhar com sistemas de robôs autônomos cooperativos em ambientes dinâmicos. Werger atribui grande parte de seu sucesso a três princípios de modelagem que desenvolveu: *minimalism*, *statelessness* e *tolerance*, nos quais se baseou para o desenvolvimento de sua arquitetura.

O minimalismo pode ser definido como a configuração mínima de recursos para realizar uma determinada tarefa com êxito. Entre seus benefícios estão a facilidade e rapidez no desenvolvimento e depuração, respostas rápidas, e a definição de sistemas executáveis mais eficientes e robustos.

Como os sistemas puramente reativos não possuem representação do ambiente em que estão inseridos, os agentes muitas vezes não conseguem garantir a continuidade de seus

comportamentos, tendendo à frequentes problemas de oscilações e perda temporária de percepção. Já nos sistemas deliberativos, onde os agentes guardam em sua memória todas as informações do ambiente no qual se encontram, é muito difícil manter em sincronia o estado do agente com o próprio ambiente, aumentando os custos e as chances de possíveis falhas. Para amenizar este problema, Werger passou a utilizar o que chamou de *statelessness*, ou seja, os agentes passaram a guardar em sua memória, por um breve período de tempo, as imagens de suas últimas percepções, não deixando, assim, de ser considerado um sistema reativo.

Segundo Werger, a tolerância pode ser obtida através da junção de *minimalism* com *statelessness*, ou seja, em caso de algum tipo de falha do agente, será muito mais fácil sua recuperação para retornar ao seu estado ideal. Isso se deve principalmente a característica de fornecer respostas rápidas, e também por possuir apenas o mínimo de recursos necessários para desenvolver determinada tarefa.

7.6.3 Arquiteturas Propostas

Baseando-se nas abordagens anteriores , sobre arquiteturas de agentes reativos, apresentaremos a seguir duas arquiteturas desenvolvidas uma para a construção de um jogador Centro-Avante , desenvolvida por [Ludwig, Cordenonsi , 2002] e outra para a construção de um jogador Zagueiro , desenvolvida por [Cordenonsi , Hammel 2001].

Os agentes foram definidos de acordo com a arquitetura de subordinação [Brooks, 1985], baseando-se na prioridade do comportamento para o estado em que o agente se encontra no ambiente. A arquitetura de subordinação foi escolhida devido a possibilidade de incrementar, gradativamente, o grau de complexidade do agente a ser desenvolvido, o que possibilitaria a inserção de novas funções.

Para a implementação dos agentes Centro-avante e Zagueiro, se fez uso de um modelo baseado em regras if-then-else com prioridades. Este esquema adapta-se bem à arquitetura de subordinação, pois cada nível da arquitetura de [Brooks, 1986] pode ser comparado a um nível de prioridade no conjunto de regras, formando uma hierarquia facilmente ajustável.

Os agentes Centro-avante e Zagueiro possuem cada um , um conjunto de regras. Cada regra possui uma pré-condição e uma ação associada. Para que uma regra seja

executada, sua pré-condição deve ser verdadeira. Caso mais de uma regra tenha sua pré-condição validada, a que tiver prioridade mais alta será executada.

7.6.3.1 Jogador Centro-Avante

O agente centro-avante aqui em questão, tem como objetivo priorizar a atitude de chutar a bola em direção ao gol adversário. A arquitetura prevista aqui, define dois jogadores centro-avantes, que cooperam entre si para atingir o objetivo maior de fazer gols.

Ao começar uma partida, o jogador deverá se encaminhar até seu ponto base, ou seja, um ponto de referência para o próprio agente e seus companheiros. O ponto base do jogador encontra-se inserido dentro de uma área de atuação que é a parte do campo da qual o agente não poderá sair, sob hipótese alguma. Após o início da partida, a área de atuação é a única parte do campo na qual o jogador poderá atuar, evitando assim, que um mesmo agente exerça mais de uma função e a aglomeração de vários agentes em um só local. A figura abaixo representa o ponto base e a área de atuação do jogador.

Para que o jogador possa saber sua localização atual dentro do campo, são utilizados alguns pontos de referência espalhados pelas linhas que formam o contorno do mesmo.

Essas referências são os *flags*, ou seja, pontos situados no ambiente que são repassados ao agente a cada ciclo executado pelo servidor. O jogador que será desenvolvido, terá como base o “*flag pl b*” ou “*flag pl t*” (veja a figura do tópico 6.5), assim ele terá condições de se posicionar adequadamente na sua área de atuação, respeitando os limites impostos durante o seu desenvolvimento. Por exemplo: após o servidor passar a mensagem com a posição (distância e ângulo) em relação ao *flag*, o agente analisa esses dados e, então, toma suas devidas decisões.

A seguir, serão analisados os principais conjuntos de condições e ações que o agente deverá respeitar ao ter posse da bola, seguindo o nível de prioridade das regras:

- o objetivo principal do agente centro-avante será chutar a bola em direção ao gol adversário, dando a esta ação a mais alta prioridade (figura abaixo);

- caso a direção ao gol esteja sendo interceptada por um adversário, o jogador deverá visualizar o companheiro mais próximo e passar a bola;
- caso a direção ao gol esteja sendo interceptada e o agente não consiga visualizar seu companheiro, deverá então passar a bola para o ponto base do mesmo.



Figura 7.6 – Representando a ação de um jogador segundo a Arquitetura de [Ludwig, Cordenonsi , 2002]

7.6.3.2 Jogador Zagueiro

Da mesma forma que o jogador centro-avante, o jogador zagueiro também possui suas prioridades tais como : proteger o seu próprio gol, dentro de limites pré-definidos, não tendo nenhuma participação em jogadas ofensivas.

Tal característica possui duas vantagens:

- cada agente, possui uma área pré-estabelecida para atuar, não interfere diretamente nas ações dos demais agentes. Este fato impede a criação de aglomerados de agentes do mesmo time em volta da bola;
- simplificação nas regras de decisão, o que influencia diretamente na velocidade de definição de ação do agente.

Para o agente zagueiro, foi definido uma área circular de raio 35 *metros* em torno do (*flag p l c*) ou (*flag p r c*) (veja a figura do tópico 6.5), que fica exatamente na frente da goleira do seu time, em cima da linha da grande área. A escolha de qual *flag* basear seu

comportamento é feita antes do início da partida, quando o simulador passa esta informação para os agentes. Todas as ações do agente zagueiro se darão dentro desta área circular. Se, por qualquer motivo, o agente afastar-se desta área, ele deverá retornar à mesma, de acordo com as regras de comportamento que estão definidas na próxima seção.

O comportamento básico para o agente, em relação à sua posição, é explicado à seguir. A cada ciclo, o agente verifica a distância entre ele e o *flag* de posicionamento.

Se a distância for superior que o raio de 35 *metros*, o agente deve retornar em direção ao *flag* até que a distância for menor que um metro. Ou seja, o agente não fica exatamente em cima do ponto, mas bem próximo dele. Esta perda de precisão é tolerável pois o posicionamento do jogador durante o jogo é bastante impreciso, pois o mesmo só é obtido através de sucessivos comandos de corrida. Por outro lado, se a distância for a menor que a definida como limite, o agente pode andar dentro da área, se estiver vendo a bola.

As regras de comportamento do agente zagueiro são descritas na tabela abaixo, em ordem decrescente do grau de prioridade.

1	SE a distância do agente até a bola for menor que 0,5 <i>metros</i>	ENTÃO o agente deve chutá-la em
2	SE a distância do agente até o <i>flag</i> de posicionamento for superior a trinta e cinco <i>metros</i>	ENTÃO o agente deve virar-se para o <i>flag</i> e correr em direção ao mesmo
3	SE o agente percebe a bola e a mesma estiver dentro de sua área	ENTÃO o agente deve correr até a bola
4	SE o agente percebe a bola, mas esta não está dentro de sua área	ENTÃO o agente deve girar em direção a bola
5	SE o agente não percebe a bola, percebe o <i>flag</i> de posicionamento e a distância deste até o agente for maior que 0,1 <i>metro</i>	ENTÃO o agente deve correr até o <i>flag</i> de posicionamento
6	SE o agente não percebe a bola, mas a distância do <i>flag</i> de posicionamento for menor que 0,1 <i>metro</i>	ENTÃO o agente deve girar em torno do seu eixo
7	SE o agente não percebe a bola nem o <i>flag</i> de posicionamento	ENTÃO o agente deve girar em torno do seu eixo

Tabela 7.2 – Representação das regras de comportamento para um agente zagueiro

O agente, quando está em condições de chutar a bola, sempre deverá chutá-la em direção ao gol adversário. Neste sentido, outro(s) agente(s) do mesmo time deverão ser implementados para que atuem na área de meio-campo, para que recebam a bola chutada pelo zagueiro. De acordo com a segunda regra, percebe-se que o agente só poderá ir de encontro a bola se a mesma estiver dentro de sua área de atuação. Caso contrário, outros agentes deverão interceptar a mesma.

Quando o agente percebe a bola em sua área de percepção, ele irá diretamente de encontro a ela. Ao diminuir a distância para menos de meio metro, ele poderá chutar a bola em direção ao gol adversário. No entanto, se o agente percebe a bola, mas esta não está dentro de sua área de atuação, o mesmo deverá girar em torno do seu eixo em direção da bola. Desta maneira, mesmo não indo de encontro a bola, ele se posicionará de frente para mesma. Esta regra diminui o tempo de resposta do agente quando este tiver que correr até a bola, pois evita-se que o mesmo deva girar primeiro e depois correr (dois ciclos de simulação). Logo, pela regra quatro, o agente deverá sempre ficar de frente para a bola enquanto a tiver no seu campo de visão.

Em relação as duas últimas regras, se o agente não perceber a bola, ele deve retornar à sua posição original, perto do *flag* de posicionamento. Caso ele já esteja perto do *flag*, ele deverá girar em torno do seu eixo para tentar encontrar a bola. Finalmente, caso o agente não esteja percebendo a bola nem o *flag*, ele deverá girar em torno do seu eixo até encontrá-los.

7.7 Principais Comportamentos dos agentes jogadores

Analisando o comportamento de cada jogador em campo pode-se dizer que todos os jogadores, exceto os goleiros, apresentam características comportamentais similares e têm como principal referencial a bola. Sua distinção entre si está apenas nas prioridades das funções mais específicas. Dentre seus comportamentos, por exemplo um jogador de meio campo possui como idéia principal, estar em controle da bola para realizar um passe, conduzir a bola ou ainda, por ventura, chutar ao gol adversário. Esse tipo de jogador prioriza comportamentos como o de chutar a bola como responsabilidade de um companheiro.

Nos subtópicos a seguir encontra-se alguns dos principais comportamentos de um agente jogador propostos por [Bagatini, Alvares, 2001].

1) Saber onde se localiza a bola

O jogador tende a localizar a bola sempre que a mesma não estiver visível. Para isso, o jogador verifica o último ângulo conhecido da bola. Logo após, ele gira no mesmo sentido. Caso não consiga determinar o último ângulo conhecido, o jogador executa um giro segundo

um ângulo determinado e verifica novamente se vê a bola, caso não veja a bola mantém girando até vê-la.

2) Passar a bola para o companheiro

Para realizar um passe, o jogador que tem o controle da bola, verifica a disponibilidade de seus companheiros (se no caso ele está desmarcado). Ele tentará chutar a bola para o companheiro mais apropriado.

Caso o jogador que tenha a posse de bola esteja marcado, ele tentará chutar para o companheiro procurando evitar o adversário que está próximo.

Para realizar esse comportamento, o jogador segue algumas pré-condições como: verificar os companheiros mais próximos dele ou do gol adversário, analisando quais estão desmarcados. Sempre antes de um passe, o jogador procura ter como referencial o gol do adversário. Essa condição procura proporcionar que o mesmo, ao realizar um passe, prefira um passe em direção ao gol do adversário e não ao seu próprio gol.

3) Interceptar a bola

A tarefa de interceptar a bola corresponde à ação do jogador de ir para um ponto da trajetória da bola, no qual ele possa ter o domínio da mesma.

O jogador procura interceptar a bola, calculando um ponto de interceptação. Esse ponto corresponde à interceptação entre duas linhas. Uma delas é a linha formada por dois pontos, que são o deslocamento da bola em dois ciclos consecutivos (o atual e o anterior), a outra é uma linha perpendicular traçada à primeira que passa pelo ponto do jogador.

Conhecido o ponto de interceptação, deve-se saber se a linha da trajetória da bola irá passar perto do jogador, para isso, realiza-se uma estimativa da distância que a bola vai percorrer até parar e da distância total da bola ao ponto de interceptação. Caso o valor dessa estimativa seja menor que a distância total da bola ao ponto, a ação não é executada, pois a bola não chegaria ao ponto. Caso contrário, o jogador calcula a direção para a qual ele vai se virar para que fique de frente para o ponto (utilizou-se o ângulo absoluto do jogador) e finalmente correr para esse ponto.

Algumas instabilidades podem ser observadas nesse comportamento, como em momentos nos quais o jogador perde a visão da bola. Conforme o jogador se aproxima do ponto de interceptação, a bola pode acabar saindo do seu cone de visão, ocasionando o comportamento de procurar a bola. Em muitos casos, o jogador abandonava a ida ao ponto de interceptação e começava a procurar a bola.

Para solucionar esse inconveniente, pode-se considerar que, se a distância do jogador ao ponto de interceptação for pequena (menor do que 8m, por exemplo), o jogador continua a sua caminhada em direção à interceptação da bola. Caso contrário, é feita uma reavaliação do ponto de interceptação, a cada nova informação visual. É permitido ao jogador que se mantenha caminhando em direção ao ponto de interceptação quando sua distância é menor do que 8m, isso, porque, quando o mesmo encontra-se a uma pequena distância da linha da trajetória da bola, é mais provável que, no ângulo de visão do jogador, não esteja aparecendo a bola.

4) Posicionando-se em campo

Tanto o meio de campo, quanto o atacante e o zagueiro, possuem uma área de posicionamento em campo. Essa área corresponde a uma região em torno da posição inicial do jogador. Essa região é limitada por um círculo imaginário, que possui como centro a posição inicial do jogador e tem como objetivo auxiliar no posicionamento do mesmo. O jogador procura manter-se em sua área de posicionamento em momentos nos quais não faz parte de uma jogada, ou seja, sempre que estiver a grande distância da bola e seu time possuir o controle da mesma. Nesse caso, o jogador irá retornar a um ponto dentro da sua área de posicionamento, que não necessariamente seja o seu ponto inicial. Estando em sua região de posicionamento, o jogador passa a executar outras ações.

O comportamento de “posicionar-se em campo” também possibilita a perda das informações da bola, visto que o jogador tende a se virar para o ponto inicial, ocasionando a perda da visão da bola, o que possibilita o comportamento de procurá-la. Para a solução desse problema pode ser incluída uma condição na regra de procurar a bola: sempre que a última ação for girar para o ponto inicial, deve-se ignorar a regra de procurar a bola e seguir para a posição. Como essa ação é utilizada quando o jogador está a uma grande distância da

jogada, não é necessário fazer a procura da bola (em seguida), logo, a ação de voltar para a região de posicionamento pode ser realizada.

5) Marcar o adversário

O problema da marcação envolve vários fatores e tal comportamento pode ser utilizado por diferentes jogadores. No entanto, as diferentes prioridades podem definir esse comportamento da seguinte forma: um zagueiro pode estar sempre procurando marcar, já um meio de campo irá marcar no momento em que nenhum outro comportamento for satisfeito.

Para o comportamento marcar acontecer, o jogador deve saber qual adversário marcar. Uma solução simples que apresenta razoável resultado é a de procurar marcar o jogador adversário mais próximo que está desmarcado.

Quando a bola está visível, considera-se a região de marcação um círculo de raio de aproximadamente 5m em volta do adversário. O jogador procura ir para uma região entre a bola e o adversário. Para achar a direção na qual o jogador irá virar-se, toma-se como referência a direção do adversário somada de um certo desvio que fará com que o jogador fique entre a bola e seu oponente.

No caso da bola não estar visível, o jogador vai em direção ao adversário. Na maior parte, esse comportamento faz com que o jogador fique atrás de seu adversário. No entanto, quando o jogador chega até o oponente, ele começa a procurar a bola e ao achar a bola, ele pode calcular a direção e se locomover para a região entre a bola e o adversário (caso ainda não esteja nela).

6) Ir para a bola

Outro comportamento atribuído ao jogador é o de ir para a bola sempre que não existir companheiro perto da bola (aproximadamente em um raio de 3m da bola). Nesse caso, o jogador considera-se o mais próximo da bola e vai em direção a ela, mesmo estando a uma grande distância. Esse comportamento também permite que o jogador possa se libertar de sua região de posicionamento, proporcionando uma maior movimentação em campo.

7) Manter-se desmarcado

A ação consiste em caminhar desviando do adversário, procurando afastar se do mesmo. O jogador tende a ir para uma direção que o deixe mais próximo da bola. No caso dele não estar enxergando a bola, atribui uma direção randômica para seu movimento. O jogador tente a realizar o desvio enquanto o adversário ainda estiver em sua área de marcação. Para executar tais tarefas, o jogador deve estar sendo marcado pelo adversário, e ter algum companheiro perto da bola. Essas condições evitam que o jogador desvie de seu objetivo principal que é o de ter o controle da bola para o seu time.

7.8 Comandos de Controle

Durante uma partida cada agente pode executar comandos (que se traduzem em ações). O servidor executa os comandos no final de cada ciclo.

7.8.1 Comandos corporais

Todos os movimentos e comportamentos dos jogadores são consultados com base numa série de comandos simples que passamos a descrever :

- **(turn Moment)**

O “Moment” é o número de graus (de -180 a 180). Este comando faz com que o jogador rode, a direção do seu corpo, “Moment” representam os graus relativos a atual direção do agente jogador.

- **(dash Power)**

Este comando acelera o jogador na direção do seu corpo. O “Power” varia entre -100 a 100.

- **(kick Power Direction)**

Este comando faz com que o jogador chute a bola com uma determinada força (“Power”) e uma determinada direção (“Direction”).

- **(catch Direction)**

Este é um comando especial que funciona apenas para o goleiro. Com este comando o goleiro tenta apanhar a bola numa determinada direção (“Direction”).

- **(move X Y)**

Este comando só pode ser usado antes do início do jogo e depois de um gol. Este comando move instantaneamente o jogador para a posição X (correspondendo ao comprimento do campo) que está no intervalo de -54 a 54 e a posição Y (correspondendo a largura do campo) que está no intervalo de -32 a 32.

- **(turn_neck_angle)**

Este comando faz com que o jogador rode o pescoço num determinado ângulo (“Angle”). O pescoço poder ser rodado dentro de um intervalo de -90 a 90.

Note-se que em cada ciclo só pode ser executado um dos comandos descritos (a exeção do turn neck que pode ser executado independente dos ciclos).

7.8.2 Comandos de Comunicação

A única maneira de transmitir mensagens entre dois jogadores é através do uso do comando say e ouvir através do sensor hear.

- **(say Message)**

Este comando faz a distribuição da mensagem através do campo para todos os jogadores que estejam perto o suficiente para ouvir (o valor da distância especificada em áudio para que os jogadores consigam se comunicar é de 50 metros).

- **(ok say)**

Significa que o comando foi bem sucedido. Em caso de erro, o servidor responde o seguinte : (error illegal command form).

7.8.3 Outros Comandos

- **(sense body)**

Este comando serve para pedir ao servidor informação do estado do seu corpo. Note-se que em versões superiores o servidor envia automaticamente esta informação a cada ciclo.

- **(score)**

Pedir ao servidor que envie informação sobre o número de gols de cada equipe. O servidor responde no seguinte formato : (score Time OurScore OpponentScore)

- **(change view Width Quality)**

Muda os parâmetros da visão do jogador.

8. O TREINADOR

8.1 Apresentação

Como na vida real, as equipes de futebol simulado também podem ter seus próprios treinadores, que os orientam durante o jogo. O treinador é o décimo segundo jogador de uma equipe, e a sua função é dar apoio aos jogadores em campo.

Existem dois tipos de treinadores, o treinador de jogo (online coach), que pode participar nos jogos e um outro treinador que não participa nos jogos (off-line coach). Ao treinador que pode participar do jogo vamos passar a chamá-lo de “treinador on-line” e o outro de “treinador de arquibancada”.

8.2 Diferença entre o Treinador On-line e o Treinador de Arquibancada

O **treinador de arquibancada** é usado nas fases de desenvolvimento em tarefas como aprendizagem ou mesmo controlar jogos treino. O **treinador online** é usado durante os jogos para dar ajuda extra aos jogadores.

Na fase de desenvolvimento de clientes, por exemplo quando se querem aplicar algoritmos de aprendizagem para que os jogadores aprendam a driblar ou efetuar passes, talvez seja necessário criar sessões de treino automáticas.

O **treinador de arquibancada** possui as seguintes capacidades :

- pode controlar o modo de jogo (kick in, corner kick, goal kick, play on);
- pode transmitir mensagens aos jogadores. Essas mensagens podem ser comandos ou qualquer outra informação , a sintaxe e interpretação definidas pelo responsável da equipe.

- pode alterar a posição dos jogadores e da bola para qualquer setor do campo, e alterar a velocidade e direção de qualquer jogador.
- recebe informação global e não distorcida de todos os objetos móveis.

O **treinador online** é um pouco mais limitado, tem as seguintes características :

- pode comunicar-se com os jogadores;
- recebe informação global e não distorcida de todos os objetos móveis.

9. CONSIDERAÇÕES FINAIS

O presente trabalho teve como objetivo estudar o ambiente de Simulação da RoboCup, através de subsídios teóricos e práticos para a compreensão do simulador.

Com o objetivo alcançado, o material servirá como contribuição significativa no sentido de colocar o departamento de informática e estatística (INE), da Universidade Federal de Santa Catarina, por dentro do assunto, esperando-se que esse seja um primeiro passo na direção de novas pesquisas, como por exemplo, a criação do primeiro time de futebol de robôs desenvolvidos pelo departamento, além é claro, servir como material de apoio à pesquisas na Área de Inteligência Artificial e Robótica.

Finalizando-se o estudo pode-se concluir que não era a intenção esgotar a temática, porém tendo em vista a carência de materiais de apoio sobre o assunto, fez-se necessário o estudo uma vez que o simulador é um ambiente que abre as portas para novas descobertas e investigações, no sentido de impulsionar novas pesquisas que venham a somar e enriquecer cada vez mais a nossa Ciências da Computação mais respectivamente na área da Inteligência Artificial e Robótica.

10. REFERÊNCIAS BIBLIOGRÁFICAS

- BAKER, Albert.** JAFMAS – A java-based agent: framework for multiagent systems. Development and Implementation. **Cincinnati: Department of Electrical & Computer Engineering and Computer Science University of Cincinnati, 1997- Doctoral thesis.**
- BAGATINI, Daniela D.S. e ALVARES, Luis Otávio C., 2001** O desenvolvimento de um Agente Goleiro para o Simulador Soccer Server.
- BARRETO, Jorge Muniz.** Inteligência Artificial no limiar do século XXI. Florianópolis : PPP Edições : Terceira Edição – Florianópolis, 2001.
- BIANCHI, R.^aC.** Uma arquitetura de Controle Distribuída para um sistema de visão computacional Propositada. São Paulo, 1998.
Disponível em <<http://www.iti.pcs.usp.br/~rbianchi/papers.html>>
- BORDINI, Rafael Heitor., VIEIRA, Renata., MOREIRA, Álvaro Freita., Fundamentos de Sistemas Multiagentes. Porto Alegre, ago. 2001.**
Disponível em <<http://www.inf.ufrgs.br/~bordini/Publications/jaii-2001/>>
- BROOKS, Rodney ^a,** A Robust Layered Control System for a Mobile Robot. **IEEE Journal of Robotics and Automatin., New York, 1986.**
- BUENO, Fancisco da Silveira.** Minidicionário da Língua Portuguesa. **São Paulo: FTD: LISA, 1996.**
- CORTEN, Emiel; DORER, Klaus; HEINTZ, Fredrik; KOSTIADIS, Kostas; KUMMENEJE, Johan; MYRITZ, Helmut; NODA, itsuki; RIEKKI, Jukka; RILEY, Patrick; STONE, Peter; YEAP, Tralvex., 1999 Soccer Server Manual Ver. 7 Ver. OO beta(for Soccer Server Ver. 5.00 and later)**
- CORDENONSI, A..Z.,HAMMEL, G.** Desenvolvimento de um agente Zagueiro para a ROBOCUP. In: **III Simpósio de Informática do Planalto Médio. UPF, Passo Fundo, Rs, 2002.**
- DEMAZEAU, y.** From interactions to collective behaviour agent-based system. In: **European Conference On Cognitive Science, I, 1995, St. Malo. Proceedings... France**
- FERBER, J.** Multi-Agent Systems: Introduction to Distributed Artificial Intelligence{s1:sn}, **1999.**

FRANKLIN, Stan e GRAESSER, Art, Is It in Agent, or Just a Program ? : A Taxonomy for Autonomous Agents, Universidade de Memphis, 1996.

FONER, L. What's in agent, anyway? A sociological case study Technical report, MIT Media Lab, 1993.

GUDWIN, Ricardo R. Introdução à Teoria de Agentes. DCA-FEEC-UNICAMP.2003.

JENNINGS, Nicholas R., A Roadmap of Agent Research and Development, 1998.

LCMI- Laboratório de Controle e Microinformática. Departamento de Automação e Sistemas, Florianópolis, out 2000.

Disponível em: <<http://www.lcmi.ufsc-team/index.htm>>

LUDWIG, G.^a, CORDENONSI, A.Z. Desenvolvimento de uma agente Centro-avante para a ROBOCUP. In: I Simpósio de Informática da Região do RS Santa Maria – RS. Anais 2002.

LUFT, Celso Pedro. Minidicionário LUFT, São Paulo. Ed. Ática, 1998.

MAYFIELD, James et. Al.. Cooperating Mobile Agentes for Mapping Networks, Maio de 1998.

MENESES, Eudenia Xavier. Jornada de Atualização em Inteligência Artificial- Integração de Agentes de Informação. 2001.

OLIVEIRA, R.C.L., de AZEVEDO, F.M.& BARRETO, J. M. Dynamic neural net in the state space utilized in non-linear process identification. In ICANGA'97- International Congress on Neural Networks and Genetic Algorithms, Norwich, 1997.

TORSUN, I.S. Foundations of Intelligent Knowledge-based systems. London: Academic Press, 1995.

VREESWIJK, Gertrud A. W., Open Protocol in Multiagent Systems, University of Limburg, 1995.

WEISS, Gerhard. Multiagent Systems: A modern approach to distributed artificial intelligence. MIT, Press, 1999.

WITTIG T., ed.: "ARCHON, an architecture for multi-agent systems", Ellis Horwood, 1992, ISBN 0-13-044462-6.

WOOLDRIDGE, Michael & Jennings, Nick. Intelligent Agents: Theory and Practice. 1994.

WOOLDRIDGE, Michael & Jennings, Nick. Agent Theories Architectures and Languages A Survey. In Eds. Intelligent Agents, Springer-Verlag, 1995.

ANEXO

Estudo Sistemático de auxílio à implementação de um time de futebol de robôs simulados no simulador RoboCup

Jan Antonio Pereira

Ciência da Computação , 2004
Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC), Brasil
jpereira@inf.ufsc.br

Resumo

O presente estudo propõem-se a recolher uma documentação que venha motivar pesquisadores na Área de Inteligência Artificial e Robótica através de um ambiente de simulação aberto a investigação e a descobertas de novas tecnologias que possivelmente venham a ser aplicadas em problemas sociais e industriais. Para alcançar o objetivo proposto, inicialmente são apresentados os diversos conceitos de agentes e sistemas multiagentes, importantes para o contexto de uma partida de futebol de robôs. A seguir descreve-se o ambiente de Simulação Soccer Server, sua arquitetura cliente-servidor, seus pré-requisitos, forma de comunicação e alguns comandos. Como resultado pode-se esperar um estudo que reúne uma documentação que espera-se ser útil para um primeiro contato entre o pesquisador interessado e o ambiente de simulação Soccer Server da RoboCup.

PALAVRAS-CHAVE: Agentes; Sistemas Multiagentes; Simulador Soccer Server; RoboCup; Futebol de Robôs.

Abstract

This study proposes to join a documentation that to motive searches in The Artificial Intelligence and Robotic through the open simulation environment, an investigation and discovery new technologies that possibility to apply in social and industrial problems. To reach proposed goal initialed showing various agents and multi-agents systems conceptions essential to the context of robot soccer game. After then, described simulation soccer server environment, client-server architecture, prerequisites, communication structure and some commands. As result, can to hope that the study meet utilities documentations for the first contact in between researchers and Soccer Server Simulation environment of RoboCup.

KEY-WORDS : Agent, Multi-agents Systems, Soccer Server Simulation, RoboCup, Robot Soccer Game.

1. Introdução

O presente artigo inicia apresentado alguns conceitos de agentes e sistemas multiagentes fundamentais no contexto de uma partida de futebol de robôs. Após analisar os principais conceitos e características de agentes, será abordada a RoboCup. Sobre ela descreve-se o ambiente de simulação, sua arquitetura, seus pré-requisitos, forma de comunicação e alguns comandos. Tendo como objetivo proporcionar aos interessados um

primeiro contato com o ambiente de simulação Soccer Server da RoboCup, servirá também como material de apoio a implementação de um time de futebol de robôs.

2. Agentes e Sistemas Multiagentes

2.1 Agentes

Encontrar um único conceito ou padrão para agentes é muito difícil, pois há várias abordagens do ponto de vista de diversos autores que pode gerar significados diferentes.

Segundo o dicionário (Bueno1996) agente é o que trata de negócio por conta alheia; causa; o promotor ; o que pratica ação; tudo o que opera;

Na definição do dicionário (LUFT 1998) um agente é uma pessoa que age por ou no lugar de outra segundo autoridade por ela outorgada - é um representante da pessoa.

Em inteligência artificial, pode ser usado o termo agente para referir-se a um processo ou tarefa computacional em que o homem constrói uma ou várias funções com a capacidade de resolver problemas, adaptando-se ao um meio, reagindo a ele e provocando mudanças neste meio. Ou seja, ser capaz de aprender e tomar decisões a partir de situações diferentes.

Dentro desse amplo campo, existem inúmeras definições para “agentes”. A seguir temos uma coleção de definições retiradas de Bianchi(1998):

- um agente é qualquer coisa que pode ser vista como percebendo seu ambiente através de sensores e agindo sobre este ambiente através de efetadores;
- agentes autônomos são sistemas computacionais que habitam algum ambiente dinâmico e complexo, percebem e atuam autonomamente neste ambiente e, fazendo isto, atingem um conjunto de objetivos ou tarefas para os quais foram projetados;
- um agente é definido como uma entidade de software persistente dedicada a um propósito específico;
- agentes inteligentes realizam continuamente três funções : percebem as condições dinâmicas em um ambiente; agem para afetar as condições do ambiente; e raciocinam para interpretar as percepções, resolver problemas, realizar inferências e determinar ações;
- um agente pode ser um sistema computacional baseado em hardware ou(mais habitualmente) em software que possui as seguintes : autonomia, habilidade social, reatividade e pró-atividade;
- agentes autônomos são sistemas capazes de ações autônomas e propositadas no mundo real.

A partir dessas definições, apesar de variadas, algumas características básicas que os agentes devem possuir podem ser definidas. Algumas das mais importantes são:

PROPRIEDADE	SIGNIFICADO
Autônomo	Controle sobre as próprias ações
Pró-atividade (orientado a objetivos)	Segue objetivos e não apenas reage a mudanças no ambiente
Reativo	Respostas assíncronas a mudanças no ambiente
Temporalmente Contínuo	E um processo executado continuamente
Comunicativo	Comunica-se com outros agentes, incluindo usuários
Aprendizado	Muda seu comportamento baseado em experiências anteriores
Móvel	Pode transportar a si próprio de uma máquina a outra
Flexível	Conjunto de ações não é pré-definido
Personalidade	Personalidade verossímil e estado emocional

Principais características dos agentes

Algumas propriedades são essenciais para uma melhor caracterização de agente. Uma delas é ser autônomo. É a autonomia que leva o agente inteligente a tomar decisões importantes para a conclusão de uma tarefa ou objetivo sem a necessidade da interferência do ser humano ou qualquer entidade. Ser capaz de agir independentemente com seu ambiente através de seus próprios sensores ou com as suas próprias percepções como objetivo de realizar alguma tarefa seja ela externa ou gerada por ele próprio.

Ligado a autonomia esta a pró-atividade, que nada mais é que a capacidade que o agente deve ter de tomar iniciativas. Eles não responde simplesmente de acordo com o meio. Têm a capacidade de mostrar comportamentos baseados em objetivos.

Reatividade é a capacidade de reagir rapidamente a modificações no ambiente, ou seja, perceber o meio e responder de modo conveniente.

As primeiras três características são o mínimo necessário para que um programa possa classificar-se como agente. As restantes dizem respeito a facilidades que podem ou não estar presentes, dependendo principalmente do ambiente de atuação do agente em ambientes complexos, como o mundo físico, exigem agentes complexos e a presença ou não de outros agentes com os quais devem se comunicar. Em particular, os requisitos da flexibilidade e aprendizado são definitivamente uma exigência para que um agente possa rotular-se inteligente. Finalmente, as questões relativas a personalidade são relevantes principalmente quando é desejável interagir com pessoas num ambiente normalmente humano, tipicamente sistemas de chat online (como ChatterBot), ou mesmo jogos de computador, criando personagens mais verossímeis (FONER 93).

2.2 Sistemas Multiagentes

Sistemas Multiagentes caracteriza-se pela existência de agentes que interajam de forma autônoma e trabalhem juntos para resolver um determinado problema ou objetivo(TORSUN 1995).

Os sistemas Multiagentes permitem modelar o comportamento de um conjunto de entidades inteligentes e organizadas de acordo com leis do tipo social. Essas entidades, ou agentes, dispõem de uma certa autonomia e estão imersos num ambiente com o qual necessitam interagir, pelo que devem possuir uma representação parcial deste ambiente e meios de percepção e comunicação.

Segundo Bordini(2001), os Sistemas Multiagentes formam uma área de pesquisa dentro da Inteligência Artificial Distribuída(IAD), que se preocupa com todos os aspectos relativos à computação distribuída em sistemas de inteligência artificial. Em Sistemas Multiagentes, o enfoque principal é prover mecanismos para a criação de sistemas computacionais a partir de entidades de software autônomas, denominadas agentes, que interagem através de um ambiente compartilhado por todos os agentes de uma sociedade, e sobre o qual estes agentes atuam, alterando o seu estado.

Diante disto, quer-se dizer que é preciso prover mecanismos para a interação e coordenação destas entidades(agentes), já que cada uma possui um conjunto de capacidades específicas, bem como possuem seus próprios objetivos em relação aos estados do ambiente que querem atingir, exatamente porque cada agente possui um conjunto específico e limitado de capacidades. Frequentemente os agentes precisam interagir para atingirem seus objetivos.

Em suma, pode-se dizer que Sistemas Multiagentes são sistemas constituídos de múltiplos agentes que interagem ou trabalham em conjunto de forma a realizar um determinado conjunto de tarefas ou objetivos. Esses objetivos podem ser comuns a todos os agentes ou não. Os agentes dentro de um sistema multiagente podem ser heterogêneos ou homogêneos, colaborativos ou competitivos, etc. Ou seja, a definição dos tipos de agentes depende da finalidade da aplicação que o sistema multiagente está inserido.

Os sistemas multiagentes com agentes reativos são constituídos por um grande número de agentes. Estes são bastante simples, não possuem inteligência ou representação de seu ambiente e interagem utilizando um comportamento de ação/reação. A inteligência surge a partir de várias trocas de interação entre os agentes e o ambiente. Ou seja, os agentes não são tão inteligentes individualmente, mas o comportamento global é.

Já os sistemas multiagentes constituídos por agentes cognitivos possuem uma quantidade bem menor de agentes cognitivos comparado aos sistemas multiagentes reativos. Estes, conforme a definição de agentes cognitivos, são inteligentes e contêm uma representação parcial de seu ambiente e dos outros agentes. Podem, portanto, comunicar-se entre si, negociar uma informação ou serviço e planejar uma ação futura. Esse planejamento de ações é possível pois em geral os agentes cognitivos são dotados de conhecimentos, competências, intenções e crenças, o que lhes permite coordenar suas ações visando a resolução de um problema ou a execução de um objetivo.

Atualmente a pesquisa em sistemas multiagentes está interessada principalmente na coordenação das ações e comportamentos dos agentes, como eles coordenam seu conhecimento, planos, objetivos e crenças com objetivo de tomar ações ou resolver problemas.

2.3 Futebol de Robôs

O futebol sem dúvida é um dos esportes mais praticados em todo o mundo, despertando o interesse de diversas pessoas em virtude do dinamismo da partida , colaboração entre jogadores, além das regras do jogo. Utilizar uma simulação desse tipo de ambiente acaba exigindo do desenvolvedor criar jogadores com características criativas, capaz de tomar iniciativas e decisões como na realidade acontece em uma partida de futebol real.

Partidas de futebol de robôs, além de serem extremamente motivantes para possibilitar o surgimento de um espírito de ciência e tecnologia nas jovens gerações,

constituem uma atividade que possibilita a realização de experimentos reais para o desenvolvimento de robôs que apresentam comportamento inteligente e que cooperam entre si para a execução de uma tarefa comum, formando assim um time. Uma partida de futebol de robôs é dinâmica e imprevisível, resultando num domínio bastante complexo, que exige o uso de sistemas com alto grau de autonomia atuando em malha fechada e em tempo real para solucioná-lo. Diversos tópicos específicos de pesquisa são oferecidos por este domínio, incluindo, entre outros : (i) completa integração entre percepção, ação e cognição num time de múltiplos agentes robóticos; (ii) definição de um conjunto de comportamentos reativos robustos para cada agente, isto é, cada agente deve ser capaz de realizar tanto ações individuais quanto colaborativa; (iii) percepção em tempo real, robusta e confiável, incluindo rastreamento de múltiplos objetos em movimento.

Este domínio de aplicação permite que diversas técnicas sejam testadas e, principalmente, comparadas. A construção de um time de futebol de robôs envolve a integração de diversas tecnologias, tais como: projeto de agentes autônomos, cooperação em sistemas multi-agentes, estratégias de aquisição de conhecimento, engenharia de sistemas de tempo real, sistemas distribuídos, reconhecimento de padrões, aprendizagem, controle de processos, etc. (LCMI, 2000).

Com a iniciativa de motivar a pesquisa e promover as áreas de Inteligência Artificial e Robótica, um grupo de cientistas japoneses desenvolveram em 1996 uma competição de robôs jogadores do esporte mais popular no mundo, o futebol, surgindo assim a RoboCup "The Robot World Cup", onde as partidas são disputadas entre jogadores artificiais.

3 . ROBOCUP - “The Robot World Cup”

A RoboCup é uma iniciativa internacional entre robôs ou simuladores que partiu da idéia de se criar uma Copa do Mundo de Robôs. Não por acaso, já que o futebol, é um dos esportes mais populares do mundo, o que ajuda a incentivar, comparar e divulgar métodos e trabalhos. Promovendo-se o desenvolvimento de tecnologias passíveis de serem aplicadas em problemas sociais e industriais.

A meta central da RoboCup é, por volta do ano de 2050, colocar em campo um time de futebol de robôs humanóides autônomos para enfrentar o time campeão da copa do mundo da FIFA segundo regulamentações da mesma.

A RoboCup possui várias categorias: sendo que destas todas, exceto uma, a categoria simulador tratada neste artigo não envolvem o uso de robôs reais. Esta exceção consiste numa plataforma cliente-servidor onde os jogadores são programas que conectam-se a um servidor central, o SoccerServer [Corten99], onde toda a partida é simulada.

3.1 Pré-requisitos

Antes de apresentarmos o Simulador RoboCup e suas funcionalidades , é necessário deixar explícito ao interessado a configuração mínima, quanto ao hardware e aos softwares, equipamentos e ferramentas estas fundamentais para iniciar o estudo e desenvolvimento de equipes de futebol de robôs, segundo a categoria Simulador da RoboCup.

3.1.1 Hardware e Softwares necessários

Hardware:

- Computador pessoal, Pentium II ou III 650 Mhz ou superior , com 128 MB de memória RAM ou mais, nesse caso , nada impede de se usar um computador inferior, apenas haverá perda de desempenho tornando o jogo mais lento);
- Placa de rede 10/100 , caso a simulação esteja em fase de desenvolvimento e por esse motivo for feita em um único computador, não há a necessidade da placa de rede .

Softwares:

- Sistemas Operacionais recomendados : SunOS 4.1.x, Solaris 2, DEC OSF/1, NEW-OS, Linux , Irix 5 e apesar de não oficial é possível utilizar X-Windows.
- Soccer Server – software servidor .
- Soccer Monitor – software monitor .

3.1.2 Protocolo de Comunicação

A comunicação entre processos de software tornou-se indispensável nos sistemas atuais. Um dos mecanismos mais utilizados atualmente para possibilitar comunicação entre aplicações é chamado socket.

Existem 2 modos de utilização de sockets : o modo orientado a conexão, que funciona sob o protocolo TCP (Transmission Control Protocol, ou protocolo de controle de transmissão), e o modo orientado a datagrama, que funciona sob o protocolo UDP (User Datagram Protocol, ou protocolo de datagrama de usuários). Os dois modos funcionam sobre o protocolo IP (Internet Protocol).

A grosso modo, um Protocolo de Comunicação é uma seqüência de dados que contém um significado, transmitido por um meio. No caso da RoboCup, o protocolo utilizado é o UDP, o meio é a rede de cabos e o interpretador são as placas de rede.

No caso do protocolo UDP, os dados transmitidos se classificam em dois tipos : dados binários e dados texto. Dados binários são dados caracteres numéricos, que não representam , inicialmente, qualquer significado para o usuário apenas uma seqüência de zeros e uns , e dados texto são dados contendo sentenças de texto, constituídas pelos caracteres que estão sendo transmitidos por esta rede .

O protocolo UDP é usado pelo Simulador RoboCup por ser de rápida transmissão (cerca de dez vezes mais rápido que o protocolo TCP). A desvantagem em relação ao TCP está quanto a garantia a respeito da chegada da informação a seu destino. Mas no caso do Simulador RoboCup, esta desvantagem não causa grandes problemas já que , mesmo jogadores de futebol no mundo real não correspondem “100 %” , as tarefas exigidas por eles.

4. SIMULADOR ROBOCUP

O Simulador da RoboCup é um sistema que simula o ambiente físico de uma partida de futebol e que permite, aos jogadores virtuais, competirem num ambiente multiagente incerto, complexo e dinâmico. O Simulador tenta disponibilizar um ambiente competitivo para a investigação em inteligência artificial, deixando a cargo dos investigadores o desenvolvimento das ações que simule seus agentes jogadores.

O Simulador RoboCup, está dividido em diversos componentes entre eles: o Servidor, o(s) Monitor(es), o(s) Cliente(s) ou melhor dizendo agentes jogadores, ferramentas de Log, entre outros.

5. SOCCER SERVER

O servidor, Soccer Server, é o “cérebro do jogo” do Simulador RoboCup, permite a competição de jogadores virtuais em um ambiente complexo e dinâmico que é executado em tempo real.

O jogo ocorre sob um sistema cliente/servidor que simula os movimentos da bola e dos jogadores, serve de intermediário na comunicação entre clientes (por exemplo, através de comandos say e hear) além de estabelecer as regras do jogo.

O Soccer Server, foi desenvolvido em C e C++, por questões de desempenho, e roda sem problemas em várias plataformas Unix, como SunOs, Solaris e Linux, entre outras. Apesar de existir versões de servidores para plataformas X-Windows estas são não oficiais.

6. SOCCER MONITOR

O Soccer Monitor é a parte do simulador que provê a interface visual da partida. Usando o monitor podemos vivenciar o jogo e controlar procedimentos. Existem vários tipos de visualizadores com características diferentes e que rodam em diversas plataformas. Tipicamente um visualizador fornece as seguintes informações:

- resultado no decorrer do jogo;
- nome das equipes;
- A posição da Bola;
- A possibilidade de gravar um jogo.

A comunicação entre o Monitor e o Servidor é feita através da comunicação do tipo UDP/IP porta 6000 (default). O servidor envia informações ao visualizador em cada ciclo (um ciclo tem um período de 100 milissegundos). É possível ligar vários visualizadores ao mesmo servidor permitindo assim que o mesmo jogo seja transmitido em vários terminais.

7. SOCCER CLIENTE

O Soccer Cliente corresponde a área onde é implementado o agente jogador de futebol. Para se criar agentes jogadores nesta área principalmente para iniciantes no assunto partindo-se do zero é uma tarefa não só desafiadora como uma “tolice”, haja vista que existe uma biblioteca de agentes simples prontos disponível no site <http://sserver.sourceforge.net>.

Após entrar na biblioteca de agentes, e baixar um agente, você pode até executá-lo, não esquecendo de primeiro executar o Soccer Server e em seguida o Soccer Monitor, para enfim executar o seu Soccer Cliente.

No Soccer Cliente, o desenvolvedor define o comportamento e as ações do agente jogador. Comportamentos e ações estas que vão deste movimentar-se pelo campo, chutar ou passar a bola até a criação de estratégias que definam funções a cada jogador, como por exemplo, a de um atacante ou zagueiro em campo.

Criar agentes jogadores não é uma tarefa nada fácil, exige do desenvolvedor conhecimentos sobre diversos assuntos, como por exemplo arquitetura de agentes, conhecer o que já foi feito e como o fazem, os comandos existentes, as limitações por se estar desenvolvendo algo em tempo-real, são passos fundamentais antes de se aventurar na implementação de uma equipe de futebol de robôs simulados.

7.1 Características de um agente jogador

Um agente jogador é dotado de algumas características tais como :

Características	Significado
Autônomo	Controla suas próprias ações
Pró-ativo (Orientado a Objetivos)	Segue objetivos e não apenas reage a mudanças no ambiente
Pode ser flexível ou não	Em relação ao conjunto de ações pré-definidas
Possui ou não a capacidade de aprender	Mudança no comportamento baseado em experiências anteriores
É Comunicativo	Comunica-se com outros agentes
Temporalmente Contínuo	É um processo executado continuamente

Tabela com as características de um agente jogador

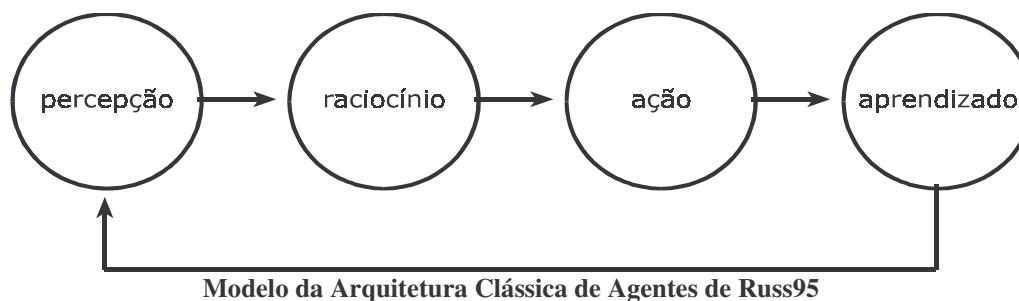
Embora muitos pesquisadores na área desenvolvam agentes jogadores de futebol, rotular um agente como inteligente exige que tal agente possua a característica de flexibilidade e capacidade de aprender, onde poucas equipes desenvolvidas até hoje chegaram.

Em sua grande maioria as equipes desenvolvidas, possuem um conjunto simples fixo de regras pré-definidas e deixam a desejar em relação a capacidade de aprender. A preocupação maior está em posicionar a equipe em campo de modo que joguem da maneira mais próxima de uma partida de futebol entre humanos, isto é, obedecendo posicionamentos, funções em campo como agarrar, defender e atacar.

7.2 Arquitetura Clássica de Agente

A arquitetura de agentes é definida por [Wooldridge, Jennings, 1994] como : “uma metodologia particular para definir agentes. Especifica como o agente pode ser decomposto na construção de um ambiente em módulos e como estes módulos podem interagir. O conjunto de módulos e suas interações devem prover uma resposta para a questão de como os sensores e o estado interno corrente do agente determinam suas ações e o futuro estado. Uma arquitetura deve prever as técnicas e algoritmos para suportar esta metodologia.”

De acordo com [Dhein, 2000], as arquiteturas de agentes são a ligação entre as especificações teóricas e a obtenção de resultados práticos, na medida em que buscam a implementação de sistemas segundo tais especificações.



7.2.1 Percepção

Para determinar qual ação tomar, um agente deve primeiro saber em que estado se encontra, para isso o agente consulta seus sensores . Os *sensores*, são as unidades que extraem do ambiente dados relevantes ao agente e os quantificam de uma maneira conveniente.

A mensagem recebida é analisada gramaticalmente através do seu “parser”. O parser tem por característica analisar a mensagem recebida, classificando os parâmetros dessa mensagem. A partir da mensagem é possível reconhecer sua natureza (visão, audição e estado do corpo) e identificar informações como : tipo do objeto, distância e direção. A figura abaixo apresenta parte de uma mensagem visual (see) recebida do servidor e a decomposição da mesma após o parser.

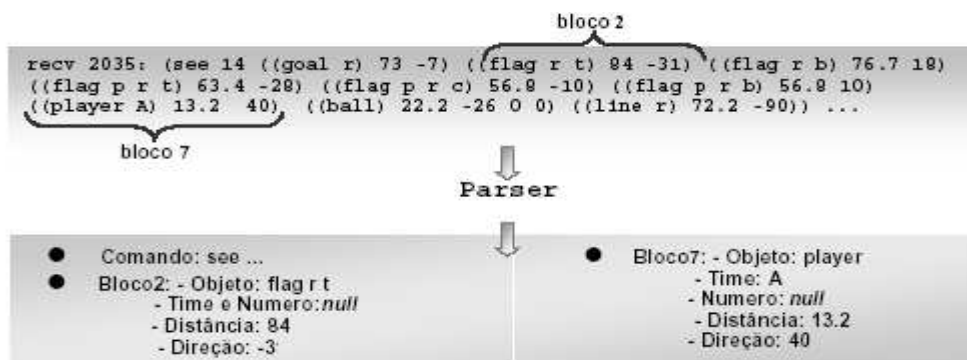


Figura ilustrativa de como as mensagens são interpretadas

As informações presentes na mensagem informam ao agente cada objeto visível por ele. O servidor tendo enviado a informação do ambiente a todos os agentes o estado do jogo é atualizado.

No caso do simulador Soccer Server, a informação visual é relativa apenas ao campo de visão do cliente, mas é recebida sob a forma de strings (na verdade, S-expressões Lisp) periodicamente, com uma frequência que é limitada por parâmetros da qualidade da informação que se deseja (no cliente) e configuração (no servidor), mas que normalmente variam entre 37.5 milissegundos e 300 milissegundos.

Também as informações sobre a propriocepção, isto é, o estado do cliente, são recebidas, mas estas últimas apenas quando o cliente indicar que assim o deseja, e não regularmente como as visuais.

7.2.1.1 Memória

O agente jogador fazendo uso da sua percepção pode criar um modelo representativo do ambiente através de uma memória. Essa memória permite ao jogador relembrar o estado do jogo nos últimos ciclos. A representação do ambiente armazenado na memória fica parecida com a figura a seguir.

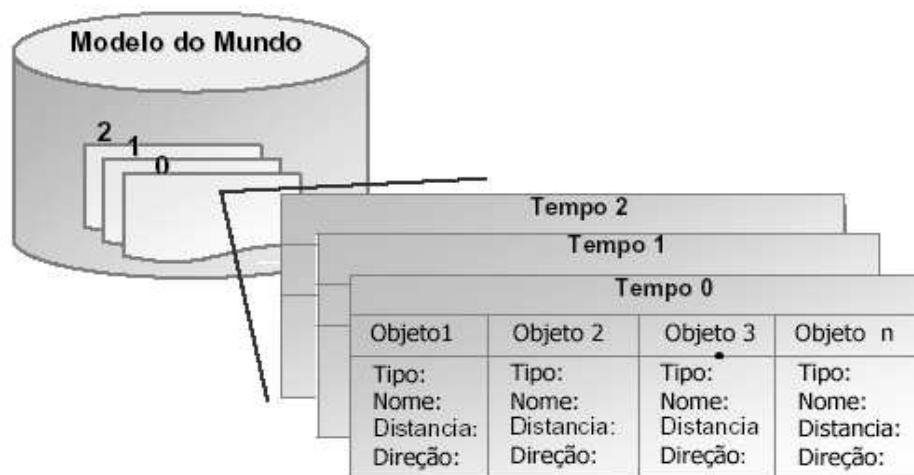


Figura 7.3 – Representação do ambiente sendo armazenado em memória pelo jogador

Essa informação é usada quando necessária, como por exemplo, se a bola não estiver visível no ciclo atual, a última informação conhecida da bola pode ser usada, ou ainda, feitas combinações entre as informações durante os três últimos ciclos que possam verificar se um jogador (companheiro ou adversário) está se aproximando da bola.

7.2.2 Raciocínio

Durante a fase de raciocínio, os dados adquiridos dos sensores serão examinados e processados, baseados no conjunto de regras, informação anterior recuperada da memória ou qualquer outro método que o agente disponha afim de determinar qual o próximo passo a ser

tomado. É justamente nesse ponto onde os agentes implementam sua “inteligência”, desde os julgamentos mais primitivos até as abstrações mais complexas. Os agentes que apresentam essa camada são ditos *cognitivos*, uma vez que chegam a uma conclusão sobre o ambiente, e conseqüentemente, sobre si próprios, baseados na análise dos dados disponíveis. Em contraste temos os agentes *reativos*, que simplesmente disparam respostas a estímulos do ambiente. Por exemplo, um robô que joga na defesa e que vê a bola em sua frente poderia simplesmente chutá-la, tentando evitar um gol do adversário. Esses agentes tendem a responder muito mais rapidamente a certas situações críticas, mas não podem planejar ações, isto é, conceber conjuntos de passos para desenvolver uma determinada intenção, o que pode limitar seriamente sua performance enquanto jogadores de futebol, em particular num ambiente onde a consciência dos outros agentes principalmente a dos companheiros é essencial para um bom desempenho.

A camada de raciocínio não é implementada em nenhum nível no servidor Soccer Server, uma vez que cada cliente deve absorver ou descartar as informações que achar necessárias, e tomar as medidas que considerar cabíveis, baseados em seu próprio conjunto de regras.

7.2.3 Ação

Após a análise das condições do agente e do seu ambiente, e baseado no seu conjunto de regras, é finalmente tomada a decisão sobre que ação deverá ser efetuada a seguir. Essa ação pode sempre incluir a possibilidade do agente continuar no estado atual, isso é, de não fazer nada. O agente atua no ambiente através dos efetadores, que exercem um papel inverso dos sensores: transformar intenções do agente em ações efetivas sobre os elementos do ambiente.

Os efetadores no simulador Soccer Server são strings ou S-expressões, que indicam o desejo do agente de realizar alguma ação no ambiente de jogo. Essas strings podem indicar que o agente deseja correr, chutar a bola, girar o corpo ou o pescoço, falar algo ou simplesmente requisitar informações sobre seu estado. Esses comandos recebem parâmetros que devem ser especificados pelos agentes, e podem falhar devido a restrições impostas sobre o ambiente pelo simulador, assim como certas ações não podem ser feitas no mundo físico, como chutar uma bola que se encontra a 10 metros de distância do agente. Cabe aos agentes determinar se é possível ou não realizar as ações desejadas na fase de raciocínio, uma vez que o servidor simplesmente ignora comandos que não possam ser efetuados.

7.2.4 Aprendizado

Alguns agentes ajustam seu conjunto de regras baseados nos resultados de suas ações sobre o ambiente. Esses agentes monitoram o ambiente antes e depois da execução de cada uma de suas ações para determinar se agiram de uma maneira adequada, e reforçam ou penalizam certos conjuntos de regras baseados nesse julgamento. Tais agentes são ditos agentes com aprendizado on-line, ou feedback on-line, e normalmente tendem a não repetir erros anteriores, uma característica certamente bastante desejável.

7.3 Comandos de Controle

Durante uma partida cada agente pode executar comandos (que se traduzem em ações). O servidor executa os comandos no final de cada ciclo.

7.3.1 Comandos corporais

Todos os movimentos e comportamentos dos jogadores são consultados com base numa série de comandos simples que passamos a descrever :

(turn Moment)

O “Moment” é o número de graus (de -180 a 180). Este comando faz com que o jogador rode, a direção do seu corpo, “Moment” representam os graus relativos a atual direção do agente jogador.

(dash Power)

Este comando acelera o jogador na direção do seu corpo. O “Power” varia entre -100 a 100.

(kick Power Direction)

Este comando faz com que o jogador chute a bola com uma determinada força (“Power”) e uma determinada direção (“Direction”).

(catch Direction)

Este é um comando especial que funciona apenas para o goleiro. Com este comando o goleiro tenta apanhar a bola numa determinada direção (“Direction”).

(move X Y)

Este comando só pode ser usado antes do início do jogo e depois de um gol. Este comando move instantaneamente o jogador para a posição X (correspondendo ao comprimento do campo) que está no intervalo de -54 a 54 e a posição Y (correspondendo a largura do campo) que está no intervalo de -32 a 32.

(turn_neck_angle)

Este comando faz com que o jogador rode o pescoço num determinado ângulo (“Angle”). O pescoço poder ser rodado dentro de um intervalo de -90 a 90.

Note-se que em cada ciclo só pode ser executado um dos comandos descritos (a exeção do turn neck que pode ser executado independente dos ciclos).

7.3.2 Comandos de Comunicação

A única maneira de transmitir mensagens entre dois jogadores é através do uso do comando say e ouvir através do sensor hear.

(say Message)

Este comando faz a distribuição da mensagem através do campo para todos os jogadores que estejam perto o suficiente para ouvir (o valor da distância especificada em áudio para que os jogadores consigam se comunicar é de 50 metros).

(ok say)

Significa que o comando foi bem sucedido. Em caso de erro, o servidor responde o seguinte : (error illegal command form).

7.3.3 Outros Comandos

(sense body)

Este comando serve para pedir ao servidor informação do estado do seu corpo. Note-se que em versões superiores o servidor envia automaticamente esta informação a cada ciclo.

(score)

Pedir ao servidor que envie informação sobre o número de gols de cada equipe. O servidor responde no seguinte formato : (score Time OurScore OpponentScore)

(change view Width Quality)

Muda os parâmetros da visão do jogador.

8. CONSIDERAÇÕES FINAIS

A elaboração deste artigo teve como objetivo fazer uma síntese do estudo realizado no trabalho de conclusão de curso, sobre o ambiente de Simulação da RoboCup, do qual foram utilizados subsídios teóricos e práticos para a compreensão do simulador.

Com o objetivo alcançado, o material servirá como contribuição significativa no sentido de colocar o departamento de informática e estatística (INE), da Universidade Federal de Santa Catarina, por dentro do assunto, esperando-se que esse seja um primeiro passo na direção de novas pesquisas, como por exemplo, a criação do primeiro time de futebol de robôs desenvolvidos pelo departamento, além é claro, servir como material de apoio à pesquisas na Área de Inteligência Artificial e Robótica.

9. REFERÊNCIAS BIBLIOGRÁFICAS

- BAKER, Albert.** JAFMAS – A java-based agent: framework for multiagent systems. Development and Implementation. **Cincinnati: Department of Electrical & Computer Engineering and Computer Science University of Cincinnati, 1997- Doctoral thesis.**
- BAGATINI, Daniela D.S. e ALVARES, Luis Otávio C., 2001** O desenvolvimento de um Agente Goleiro para o Simulador Soccer Server.
- BARRETO, Jorge Muniz.** Inteligência Artificial no limiar do século XXI. Florianópolis : PPP Edições : Terceira Edição – Florianópolis, 2001.
- BIANCHI, R.^aC.** Uma arquitetura de Controle Distribuída para um sistema de visão computacional Propositada. São Paulo, 1998.
Disponível em <<http://www.iti.pcs.usp.br/~rbianchi/papers.html>>
- BORDINI, Rafael Heitor., VIEIRA, Renata., MOREIRA, Álvaro Freita., Fundamentos de Sistemas Multiagentes. Porto Alegre, ago. 2001.**
Disponível em <<http://www. Inf.ufrgs.br/~bordini/Publications/ jaii-2001/>>
- BROOKS, Rodney ^a,** A Robust Layered Control System for a Mobile Robot. **IEEE Journal of Robotics and Automatin., New York, 1986.**
- BUENO, Fancisco da Silveira.** Minidicionário da Língua Portuguesa. **São Paulo: FTD: LISA, 1996.**
- CORTEN, Emiel; DORER, Klaus; HEINTZ, Fredrik; KOSTIADIS, Kostas; KUMMENEJE, Johan; MYRITZ, Helmut; NODA, itsuki; RIEKKI, Jukka; RILEY, Patrick; STONE, Peter; YEAP, Tralvex., 1999 Soccer Server Manual Ver. 7 Ver. OO beta(for Soccer Server Ver. 5.00 and later)**

CORDENONSI, A.Z.,HAMMEL, G. Desenvolvimento de um agente Zagueiro para a ROBOCUP. In: **III Simpósio de Informática do Planalto Médio. UPF, Passo Fundo, Rs, 2002.**

DEMAZEAU, y. From interactions to collective behaviour agent-based system. In: **European Conference On Cognitive Science, I., 1995, St. Malo. Proceedings... France**

FERBER, J. Multi-Agent Systems: Introduction to Distributed Artificial Intelligence{s1:sn}, **1999.**

FRANKLIN, Stan e GRAESSER, Art, Is It in Agent, or Just a Program ? : A Taxonomy for Autonomous Agents, **Universidade de Memphis, 1996.**

FONER, L. What's in agent, anyway? A sociological case study **Technical report, MIT Media Lah, 1993.**

GUDWIN, Ricardo R. Introdução à Teoria de Agentes. **DCA-FEEC-UNICAMP.2003.**

JENNINGS, Nicholas R., A Roadmap of Agent Research and Development, **1998.**

LCMI- Laboratório de Controle e Microinformática. Departamento de Automação e Sistemas, **Florianópolis, out 2000.**

Disponível em: <<http://www.Icmi.ufsc-team/index.htm>>

LUDWIG, G.^a, CORDENONSI,A.Z. Desenvolvimento de uma agente Centro-avante para a ROBOCUP. In: **I Simpósio de Informática da Região do RS Santa Maria – RS. Anais 2002.**

LUFT, Celso Pedro. Minidicionário LUFT, **São Paulo. Ed. Ática, 1998.**

MAYFIELD, James et. Al.. Cooperating Mobile Agentes for Mapping Networks, **Maio de 1998.**

MENESES, Eudenia Xavier. Jornada de Atualização em Inteligência Artificial- Integração de Agentes de Informação. **2001.**

OLIVEIRA, R.C.L., de AZEVEDO, F.M.& BARRETO, J. M. Dynamic neural net in the state space utilized in non-linear process identification. In **ICANGA'97- International Congress on Neural Networks and Genetic Algorithms, Norwich, 1997.**

TORSUN, I.S. Foundations of Intelligent Knowledge-based systems.**London: Academic Press, 1995.**

VREESWIJK, Gertrud A. W., Open Protocol in Multiagent Systems, University of Limburg, **1995.**

WEISS, Gerhard. Multiagent Systems: A modern approach to distributed artificial intelligence. MIT, Press, 1999.

WITTIG T., ed.: "ARCHON, an architecture for multi-agent systems", Ellis Horwood, 1992, ISBN 0-13-044462-6.

WOOLDRIDGE, Michael & Jennings, Nick. Intelligent Agents: Theory and Practice. 1994.

WOOLDRIDGE, Michael & Jennings, Nick. Agent Theories Architectures and Languages A Survey. In Eds. Intelligent Agents, Springer-Verlag, 1995.