CORDENONSI, A. Z.; LUDWIG, G. Modelagem de Um Agente Centro-Avante Para A Robocup. In: I Simpósio de Informática da Região Centro do RS, 2002, Santa Maria. Anais do I Simpósio de Informática da Região Centro do RS. Santa Maria : Centro Universitário Franciscano, 2002.

MODELAGEM DE UM AGENTE CENTRO-AVANTE PARA A *ROBOCUP*

Glauco A. Ludwig, Andre Zanki Cordenonsi
Centro Universitário Franciscano (UNIFRA) Rua dos Andradas - 1614
glauco.l@bol.com.br, andrezc@unifra.br

Abstract: *The main objective of this paper is the development of a centre-forward agent for the RoboCup, which is characterized by soccer matches between robots and simulators. The player was based on reactive agents and can act in a dynamic environment and in an autonomous form. Using predetermined rules, its principal aim is to score goals and cooperate with his partners to reach the victory together.*

Resumo: *O projeto apresentado neste artigo tem como objetivo principal o desenvolvimento de um agente centro-avante para a RoboCup, que se caracteriza por promover partidas de futebol entre robôs e simuladores. O jogador irá se basear nas características dos agentes reativos, sendo capaz de atuar em um ambiente dinâmico e agir de forma autônoma. Através de regras pré-definidas, sua finalidade principal será marcar gols, de forma a colaborar com seus companheiros para que juntos possam alcançar a vitória.*

1 Introdução

Dentro de um contexto geral, o presente trabalho foi dividido em etapas bem definidas. Inicialmente, é apresentado um estudo sobre a área de Sistemas Multiagentes e agentes dentro do domínio da robótica. Após analisado os principais conceitos e características de agentes, será abordada a *RoboCup*. Sobre ela são mencionadas suas principais divisões, em uma das quais se encontra o *Soccerserver*, que se trata do simulador de jogos de futebol utilizado para a realização deste trabalho. Para obter um melhor aproveitamento do simulador, serão também estudadas duas arquiteturas utilizadas em robótica: a arquitetura de subordinação [Brooks, 1985] e a arquitetura desenvolvida por Werger [Werger, 1999], as quais servirão de base para a construção do agente centro-avante. Finalmente, será dado ênfase ao desenvolvimento do agente. Será tratado o modo que o mesmo deverá se comportar, como deverá agir à uma determinada situação e a maneira que será utilizada para a sua implementação.

2 Agentes

A área de Sistemas Multiagentes (SMA) se interessa pelo estudo de agentes autônomos em um universo Multiagentes [Demazeau, Muller, 1990]. Pesquisadores e estudiosos da área de Inteligência Artificial Distribuída (IAD) têm buscado soluções para aproximar cada vez mais o mundo computacional do mundo real. Entre uma destas possíveis soluções surgiu o conceito de "agente".

Devido a abrangência do termo, existem muitas definições e pouca concordância entre pesquisadores na tentativa de conceituar "agente". Portanto, para se chegar a um consenso, deve-se capturar a essência do que seria um agente e então adicionar restrições para tentar delimitar ou definir classes particulares de agentes [Franklin, Graesser, 1997].

Um conceito bastante interessante e abrangente de agente é o apresentado por [Franklin, Graesser, 1997]: "u*m agente autônomo é um sistema situado em um meio ambiente, do*

qual ele faz parte e percebe, agindo sobre o mesmo continuamente, num período de tempo, buscando suas ações em sua própria agenda, podendo suas ações afetar suas percepções futuras."

Um sistema de agentes é semelhante a uma sociedade de seres que resolvem problemas e trocam informações. De acordo com [Wooldridge, Jennings, 1994], o termo "agente" é usado para denotar um *hardware* ou *software* que possui as seguintes características:

- **Autonomia:** segundo [Davidsson, 1995], um agente autônomo é "*um sistema capaz de interagir independentemente e efetivamente com seu ambiente através de seus próprios sensores, com o objetivo de realizar alguma tarefa externa ou gerada por ele próprio.*";
- **Habilidade Social:** diz respeito à capacidade dos agentes interagir com outros agentes através de algum tipo de linguagem de comunicação [Genesereth, 1994];
- **Reatividade:** os agentes percebem o ambiente e respondem a mudanças que ali ocorrem;
- **Pro-Atividade:** são capazes de tomar iniciativas, exibindo comportamentos dirigidos ao objetivo.

Segundo [Cordenonsi, Alvares, 2000], outras características importantes dos agentes são as capacidades de comunicação e cooperação que podem haver entre eles. A comunicação é necessária para que haja troca de informações e mensagens entre os agentes, fazendo com que possam agir e tomar as devidas decisões. A cooperação é a capacidade de trabalhar em conjunto, de forma que cada agente possa atingir seus objetivos, porém tendo a participação de outro agente. Essa interação (agente/agente ou agente/ambiente), pode ocorrer de forma direta (comunicação explícita) ou de forma indireta (emissão de sinais através do ambiente).

Os agentes podem, desta forma, existir em sociedades povoadas por vários agentes, cada qual cumprindo suas tarefas, todos em busca de um único objetivo. Um exemplo é o formigueiro. Ele pode ser visto como um agente, mas também pode ser considerado como uma sociedade de agentes, onde cada agente-formiga faz sua parte no objetivo comum de manter esta sociedade ativa. Portanto, cabe a cada organização definir e impor suas regras para sua sociedade, para que os agentes possam realizar suas tarefas de acordo com o pré-definido.

De acordo com a literatura, os SMA podem ser agrupados em duas categorias: Sistemas Multiagentes Reativos e Sistemas Multiagentes Cognitivos. Para a realização deste trabalho, somente os agentes reativos serão abordados.

2.1 Agentes Reativos (AR)

Os agentes reativos, normalmente, vivem em comunidades numerosas, podendo ultrapassar a casa de milhões de agentes. Reagem a mudanças em seu ambiente ou a mensagens de outro agentes (estímulo-resposta) e são considerados muito simples, não possuindo representação de seu ambiente.

Segundo [Ferber, 1991], as principais características dos Sistemas Multiagentes reativos são as seguintes:

- não há representação explícita de conhecimento: o conhecimento dos agentes é implícito (as suas regras de comportamento) e sua manifestação se externa através do seu comportamento e dos demais agentes;
- não há representação do ambiente: o comportamento (resposta) de cada agente é baseado no que ele percebe (estímulo) a cada instante. Não há uma representação interna explícita do ambiente;

- não há memória das ações: os agentes reativos não mantém nenhum tipo de histórico de suas ações, ou seja, o resultado de uma determinada ação passada não influencia diretamente na decisão de uma ação futura;
- organização etológica: a forma de organização dos SMA reativos é similar à observada por animais que vivem em grandes comunidades;
- grande número de membros: em geral, os SMA reativos possuem um grande número de agentes, com populações que podem chegar à ordem de milhares de membros.

3 *RoboCup*

A *RoboCup* (*The Robot World Cup Initiative*) [Corten et al, 1999], surgiu da iniciativa de um grupo internacional de pesquisadores da área de Inteligência Artificial e Robótica. Seu objetivo principal é promover a competição entre robôs ou simuladores num jogo de futebol, onde várias tecnologias podem ser testadas e aplicadas para tentar realizar as tarefas desejadas com êxito.

A *RoboCup* é dividida em três categorias: *small size league*, disputada por robôs físicos de pequeno porte; *middle size league*, disputada por robôs de tamanho médio; e ainda o *Soccerserver*, disputada por agentes simuladores desenvolvidos por pesquisadores da área de SMA. Para a realização deste trabalho, será dada ênfase ao estudo do simulador *Soccerserver*, o qual pode ser utilizado para testes de pesquisas na área de SMA.

3.1 *SoccerServer*

O simulador *Soccerserver* permite a competição de jogadores virtuais em um ambiente complexo e dinâmico que é executado em tempo real. O jogo ocorre através de um sistema cliente/servidor, no qual o *Soccerserver* é formado por dois programas: o próprio *soccerserver*, que simula os movimentos da bola e dos jogadores, faz a comunicação com os clientes e estabelece as regras; e o *soccermonitor*, que fornece o campo virtual do *soccerserver*.

A comunicação entre o servidor e cada um dos vinte e dois clientes possíveis em uma partida ocorre através de *sockets*[1] UDP/IP. Utilizando um destes *sockets*, o cliente envia comandos para controlar seu jogador e recebe informações visuais e auditivas do servidor. Cada cliente controla os movimentos de um único jogador.

Alguns dos comandos mais utilizados pelos clientes são: *turn, dash, kick, change_view* e *move*. O comando *move* é utilizado para posicionar o jogador antes do início da partida. Os comandos *turn, dash, kick* e *change_view* são usados para controlar as atitudes dos jogadores durante o jogo. Os jogadores também são capazes de interagir com o meio através de três diferentes informações sensoriais: as informações visuais, auditivas e aquelas do próprio corpo, representadas respectivamente pelas mensagens *see, hear* e *sense_body*.

Cada time é composto por onze agentes jogadores, onde cada agente executa uma determinada tarefa. Como se trata de um SMA, cada agente possui um determinado objetivo, que pode ser diferenciado dos de mais. Mesmo assim, todos trabalham em busca do objetivo comum, ou seja, a vitória.

O simulador *Soccerserver* foi escrito nas linguagens de programação C e C++, e roda sem problemas em varias plataformas Unix, como SunOS, Solaris e Linux, entre outras. Já para o desenvolvimento dos clientes, qualquer plataforma capaz de se comunicar com

---

[1] *sockets* são portas de comunicação entre dois programas

o servidor pode ser utilizada, e os mesmos podem ser projetados nas mais diversas linguagens de programação, como C, C++, Java ou qualquer outra.
O programa cliente define as ações do jogador e pode ser projetado de várias formas, procurando atender as necessidades desejadas para cada agente. Esse cérebro é também chamado de arquitetura na qual o jogador foi implementado. A seguir, serão estudadas duas arquiteturas reativas que podem ser utilizadas em SMA e, baseando-se nestas, será definida uma nova arquitetura para a implementação de um agente centro-avante para a *RoboCup*.

4 Arquiteturas de Agentes

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 de módulos componentes e como estes módulos podem interagir. O conjunto de módulos e suas interações devem prover uma resposta para a questão de como os sensores de dados e o estado interno corrente do agente determinam suas ações e futuro estado interno. 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.
As arquiteturas de agentes podem ser classificadas em três diferentes abordagens: clássicas ou deliberativas; alternativas ou reativas e híbridas. Para realização deste trabalho, será dada ênfase ao estudo das abordagens reativas, as quais podem ser utilizadas para a construção de agentes jogadores para a Robocup.

4.1 Arquiteturas Reativas

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.

4.1.1 *Subsumption Architecture* (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].

4.1.2 *Cooperation without Deliberation* (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 informacões 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.

5 Arquitetura Desenvolvida

Baseando-se nas abordagens anteriormente analisadas, será desenvolvida uma arquitetura para a construção de um agente centro-avante para a *Robocup*. O agente deverá analizar de forma rápida e constante as tarefas às quais foi submetido, adaptando-se ao meio ao qual foi inserido e agindo de forma autônoma. Esses foram os principais motivos de se optar por trabalhar com conceitos de abordagens reativas, tendo em vista a simplicidade que os agentes possuem para realizar suas tarefas mais básicas com êxito, no intervalo de tempo adequado.
Para realizar a comunicação, o agente recebe mensagens do *Soccerserver* (percepções). Estas são analisadas e, após tomadas as devidas decisões, são enviadas mensagens como respostas para o servidor (ações). A partir daí, passam a ser armazenadas em um módulo que atualiza o estado da partida. A tarefa a ser efetuada pelo agente foi definida de

acordo com a arquitetura de sobordinação [Brooks, 1985], baseando-se na prioridade do comportamento para o estado em que o agente se encontra no mundo corrente.
A arquitetura será desenvolvida sobre a plataforma Linux, utilizando a linguagem de programação C, devido a velocidade que a mesma apresenta na interpretação dos códigos do programa. É importante salientar que, devido ao dinamismo do ambiente, o processamento das informações torna-se limitado por restrições de tempo, ou seja, é necessário que o agente atue de maneira adequada num determinado limite de tempo, evitando que o mesmo perca a oportunidade de agir.

5.1 Protótipo do Agente

Todos os agentes ou jogadores apresentam características comuns e possuem como referencial a bola, mas também podem apresentar as mais difentes funções. Para que cada jogador exerça suas ações de maneira correta, utiliza-se um sistema baseado em regras, que são formadas por um conjunto de condições e ações. Assim, o comportamento do agente vai depender das regras que a ele foram pré-definidas. As regras possuem ainda um nível de prioridade, como por exemplo: um agente zagueiro deve ter como prioridade chutar a bola em direção ao campo adversário, afastando a mesma da sua área de defesa. Para realização deste projeto, será desenvolvido um agente centro-avante que priorizará a atitude de chutar a bola em direção ao gol adversário. Esta arquitetura será utilizada para definir dois jogadores centro-avantes, que cooperam entre si para atingir o objetivo maior de fazer gols. Deste modo, um centro-avante sempre joga ao lado de outro jogador, definido aqui como *companheiro*.
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 1 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*", 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 2);
- 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 vizualizar seu companheiro, deverá então passar a bola para o ponto base do mesmo.

Figura 1. Ponto base e área de atuação do agente.

Figura 2. Agente centro-avante chutando em direção ao gol

O algoritmo completo das decisões que o agente deverá tomar pode ser vizualizado na figura 3. Deve-se levar em consideração que o modelo de agente descrito servirá como base para o desenvolvimento da arquitetura, ou seja, o mesmo poderá sofrer alterações no decorrer de sua implementação, de acordo com as necessidades dos desenvolvedores.

```
Posicionar no início;
Início do tempo de jogo;
Ir até o ponto base;
Enquanto em jogo
      Se vejo bola
            Se bola na área de atuação
                  Ir até a bola;
                  Girar em direção ao gol;
                  Senão tiver adversário interceptando
                        Chutar para o gol;
                  Senão
                        Se vejo companheiro
                              Passar a bola para o companheiro;
                        Senão, passar para o ponto base do companheiro;
            Senão
                  Se estou no ponto base
                        Girar em direção a bola;
                  Senão
                        Girar para o ponto base;
                        Correr para o ponto base;
      Senão, girar 15º graus;
Até fim do tempo de jogo.
```

Figura 3. Algoritmo dos comportamentos do agente cento-avante.

6 Conclusões

Este trabalho teve como objetivo fazer um estudo para o desenvolvimento de um agente centro-avante para a *Robocup*. A *Robocup* tem se tornado um ambiente bastante popular para a realização de pesquisas na área de Sistemas Multiagentes, tendo como foco realizar competições através de partidas de futebol. Fazendo parte da *Robocup* encontra-se o *Soccerserver*, um simulador que tenta assemelhar-se com o mundo real, e que fornece um ambiente dinâmico e complexo para a realização de testes e competições entre agentes simuladores.

O jogador (agente) projetado baseia-se nas características de agentes reativos, que apresentam uma estrutura interna bastante simples, não possuindo nenhuma representação do ambiente no qual estão inseridos. Para o sucesso do presente trabalho foi necessário, também, analisar os princípios das arquiteturas existentes na área da robótica, dando ênfase as abordagens reativas.

Concluindo, foi desenvolvido um modelo de arquitetura reativa que se baseou nas teorias de Brooks [Brooks, 1985] e Werger [Werger, 1998]. Esta arquitetura atua na construção de um jogador centro-avante que tem como objetivo principal marcar gols. O jogador deverá seguir regras, baseadas em um conjunto de condições e ações, logo, o comportamento do agente vai depender das regras que a ele foram pré-definidas.

7 Bibliografia

BAGATINE, Daniela D. S. e ALVARES, Luis Otávio C., 2001 **O Desenvolvimento de um Agente Goleiro para o Simulador Soccerserver.**

BROOKS, Rodney A**.,** 1985 **A Robust Layered Control System for a Mobile Robot.** IEEE Journal of Robotics and Automation.

CORDENONSI, André Z., ALVARES, Luis O., 2000 **Um ambiente de evolução de comportamentos para sistemas multiagentes reativos.** Dissertação de Mestrado. Curso de Pós-Graduação em Computação. UFRGS. Porto Alegre.

CORTEN, Emiel; DORER, Klaus; HEINTZ, Fredrik; KOSTIADIS, Kostas; KUMMENEJE, Johan; MYRITZ, Helmut; NODA, Itsuki; RIEKKI, Jukka; RILEY, Patrick; STONE, Peter; YEAP, Tralvex., 1999 **Soccerserver Manual Ver. 5 Rev. 00 beta** (for Soccerserver Ver.5.00 and later).

DAVIDSON, Paul., 1995 **On the Concept of the Concept in the Context of Autonomous Agents**. II World Conference on the Fundamentals of Artificial Intelligence

DEMAZEAU, Y. e MULLER, J. P., 1990 **Decentralized artificial intelligence.** In: Demazeau, Y. e Muller, J. P. eds. Decentralized A I., Amsterdam, Elsevier.

DHEIN, Guilherme., 2000 **Integrando Deliberação e Reatividade em uma arquitetura de agentes híbrida homogênea.** Dissertação apresentada como requisito parcial à obtenção do grau de Mestre.

FERBER, Jacques e Jacopin, Eric., 1991 **The Framework of Eco-Problem Solving**. In: Demazeau, Yves; Muller, Jean-Pierre (Eds.). Decentralized A.I..2

FRANKLIN, Stan e Graesser, 1996 **Art. Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents.** Disponível na Internet em: http://www.msci.memphis.edu/~franklin/AgentProg.html/. Data de busca: 02/04/2002

GENESERETH , M. R. and Ketchpel, S. P., 1994 **Softwares Agents**. Communications of the ACM.

HAMMEL, Gilberto e CORDENONSI, Z. André, 2001 **Implementação de um agente zagueiro para a robocup.** Trabalho final de graduação do Curso de Sistemas de Informação do Centro Universitário Franciscano

WERGER, Barry B., 1999 **Cooperation without deliberation: A minimal behavior-based approach to multi-robot teams.** Ullanta Performance Robotics and Interaction Lab. Department of Computer Science.

WOOLDRIDGE, M. And JENNINGS, N. R., 1994 **Intelligent Agents: Theory and Practice**. Knowledge Engineering Review.