**LEONARDO CARDOSO SAPUCAIA**

# Desenvolvimento de um Sistema Multiagentes: O Time UESBots de Futebol de Robôs da UESB

Vitória da Conquista – Bahia
2011

**UNIVERSIDADE ESTADUAL DO SUDOESTE DA BAHIA**
**DEPARTAMENTO DE CIÊNCIAS EXATAS - DCE**

**LEONARDO CARDOSO SAPUCAIA**

# Desenvolvimento de um Sistema Multiagentes: O Time UESBots de Futebol de Robôs da UESB

Monografia de conclusão de curso apresentada à Universidade Estadual do Sudoeste da Bahia, como requisito parcial à obtenção do título de Bacharel em Ciência da Computação.

**Orientador: Prof. Dr. Fábio Moura Pereira**

Vitória da Conquista – Bahia
2011

**LEONARDO CARDOSO SAPUCAIA**

# Desenvolvimento de um Sistema Multiagentes: O Time UESBots de Futebol de Robôs da UESB

Monografia de conclusão de curso apresentada à Universidade Estadual do Sudoeste da Bahia, como requisito parcial à obtenção do título de Bacharel em Ciência da Computação.

**Banca Examinadora**

**Orientador**:

_____

Prof. Dr. Fábio Moura Pereira
UESB – Universidade Estadual do Sudoeste da Bahia

**Membros**:

_____

Prof. Me. Camilo Alves Carvalho
IFBA – Instituto Federal de Educação Ciência e Tecnologia da Bahia

_____

Prof. Dr. Hélio Lopes do Santos
UESB – Universidade Estadual do Sudoeste da Bahia

*Eu penso. E esqueço.*
*Eu vejo. E recordo.*
*Eu faço. E compreendo.*

Confúcio

# Agradecimentos

Agradeço ao tempo pelos ensinamentos.

A minha família pelo amor, compreensão e compasso moral que me guiam durante toda a vida.

À sempre graciosa Deyse Sande por suas palavras de incentivo, conforto e carinho.

Às professoras Cátia Khouri e Vanessa Bittencourt (in memoriam) que souberam me incentivar, cada uma ao seu tempo, para que eu alcançasse e ampliasse meus limites.

Ao professor Fábio Moura, por toda orientação, motivação, apoio, paciência, dedicação, por acreditar e investir no trabalho de seus alunos.

A todos que de alguma forma contribuíram para realização desse trabalho.

# Resumo

Este trabalho descreve o desenvolvimento e funcionamento do time UESBots, criado para participar das competições de simulação de futebol em duas dimensões da Robot World Cup (RoboCup). A RoboCup é uma organização sem fins lucrativos de pesquisadores de aproximadamente 40 países que tem como missão promover progressos científicos e técnicos nos campos de Robótica e Inteligência Artificial. O popular jogo de futebol é utilizado porque provê tanto desafios científicos quanto a possibilidade de uma avaliação concreta baseada em um problema bem definido: Os agentes jogadores de um mesmo time devem trabalhar em conjunto para vencer a partida. Nos campeonatos, times de várias instituições são colocados para disputar partidas entre si, testando assim algoritmos de inteligência artificial, criando novos desafios, bem como criando novas técnicas e tecnologias para superá-los. O objetivo deste trabalho é criar um Sistema Multiagentes com coordenação de agentes cooperativos que funcione como um time de "robôs" que tomem decisões e colaborem entre si. As simulações são constituídas de 11 jogadores como num time de futebol real. Os agentes desenvolvidos possuem arquitetura em camadas verticais que combinam lógica reativa e deliberativa, arquitetura híbrida e desenvolvida em linguagem de alto nível Python.

***Palavras–Chave:*** *Sistemas Multiagentes, Agentes Baseado em Componentes, RoboCup Soccer Simulation 2D.*

# Abstract

This work describes the development and operation of the UESBots Team created to join the Robot World Cup (RoboCup) soccer competitions of 2D simulation. RoboCup is a nonprofit organization of researchers from 40 countries whose mission is to promote scientific and technical advances in Robotics and Artificial Intelligence. Popular, the soccer game is used because provides scientific challenges, and the possibility of a practical assessment based on a well-defined problem: agents of the same team must work together to win the match. In the championships, teams of many institutions compete with each other, testing artificial intelligence algorithms, creating new challenges and creating new techniques and technologies to overcome them. The main objective is to create a Multiagent System with cooperative coordination that work as a team of "bots" to make decisions and collaborate each other. The simulations consist in 11 players like a real football team. The agents developed have a vertical layered architecture that combines reactive and deliberative logic, hybrid architecture and implemented with a language Python.

***Keywords:*** *Multiagents System, Components Based Agents, RoboCup Soccer Simulation 2D.*

# Lista de Figuras

# Lista de Tabelas

# Lista de Quadros

# Lista de Abreviaturas e Siglas

**FIRA** Federation of International Robot-Soccer Association

**IA** Inteligência Artificial

**IAD** Inteligência Artificial Distribuída

**RDP** Resolução Distribuída de Problemas

**SMA** Sistema Multiagentes

**STR** Sistema em Tempo Real

**UDP** User Datagram Protocol

# Sumário

# 1. Introdução

Na obra “Eu, Robô” do escritor ficcionista russo Isaac Asimov um personagem faz os seguintes questionamentos: “Quando um sistema de percepção poderá ser chamado de consciência? Quando calcular probabilidades será tão parecido quanto pensar? Quando uma simulação de personalidade se tornará o doloroso átomo de uma alma?” (ASIMOV, 1950). De fato, já há muito tempo, um dos objetivos mais perseguidos pelo ser humano é dotar uma máquina, construída pelo mesmo, com características inteligentes.

Apesar da idéia de construir máquinas com capacidade para reproduzir a forma humana de perceber, raciocinar e agir ser algo antigo na história, os avanços recentes no âmbito da informática e da computação tornaram este ideal mais palpável, culminando no surgimento de uma área de pesquisa que propõe construir sistemas que imitem nossa capacidade de raciocinar, de perceber o mundo e identificar objetos que estão à nossa volta, tomar decisões e resolver problemas, enfim, ser inteligente. Esta área é chamada Inteligência Artificial (IA).

Desde o início da década de 1970, os estudos que tentam resolver problemas de modo cooperativo vêm crescendo rapidamente dentro da área de IA. Desde então, surgiu o termo Inteligência Artificial Distribuída (IAD), uma subárea da IA, na qual são combinadas as funcionalidades dos sistemas distribuídos com técnicas de IA.

Enquanto a Inteligência Artificial clássica tem ênfase na representação do conhecimento e métodos de inferências, a IAD baseia seu modelo de inteligência no comportamento social (cooperação), sendo a ênfase transposta para as ações e interações entre entidades denominadas agentes (SOUSA, 2007 apud ROCHA, 2003).

De acordo com RIBEIRO (2010), nos modelos de agentes inteligentes os elementos autônomos são dotados de alguma forma de inteligência pré-definida que juntamente com sensores capazes de perceber seu ambiente e as mudanças ocorridas através da intervenção humana ou de outros agentes, é capaz de tomar decisões e agir por intermédio de atuadores. Em ambientes constituídos por mais de um agente, denominado Sistemas Multiagentes (SMA), é importante que uma linguagem comum seja utilizada a fim de manter o cooperativismo. As pesquisas em sistemas multiagentes têm se tornado uma das principais tendências no domínio da IA.

O conceito de futebol de robôs foi introduzido em 1993 por Itsuki Nota (KITANO et al., 1995). A competição de futebol de robôs tem como pretexto o desenvolvimento de técnicas de IA, para aplicações robóticas, utilizando um problema padrão em que um grande leque de possíveis tecnologias possam ser examinadas.

A utilização de SMA em futebol de robôs é ideal por oferecer uma solução mais natural e eficiente. É, basicamente, um ambiente povoado por vários agentes onde cada agente pode possuir um objetivo e ações particulares. Quando vários agentes possuem um objetivo global comum, estes são denominados equipe ou time, passando a pertencer a uma sociedade (SILVA, 2006). Eles agem em conjunto com intenção de fazer gols utilizando táticas de equipe e avançam sobre os membros adversários tentando retardar suas jogadas.

A RoboCup é um projeto organizado pela Federação Internacional RoboCup com o apoio de comitês nacionais de vários países. Sua missão é promover progressos científicos e técnicos nos campos de Robótica e Inteligência Artificial e tem como objetivo final criar um time composto por robôs humanóides completamente autônomos que tenha chances reais de vencer a seleção campeã da Copa do Mundo de Futebol até o ano de 2050. Ao mesmo tempo em que o âmbito da RoboCup aumentou desde o princípio em 1997, as principais atividades permanecem e se subdividem em ligas, cada uma explorando aspectos de desenvolvimento tecnológico.

Num esforço de desenvolver conhecimento em Inteligência Artificial e em Robótica o foco deste trabalho foi a criação um time competitivo da Universidade Estadual do Sudoeste da Bahia (UESB), então chamado UESBots, para participar da RoboCup, na liga de futebol simulado em duas dimensões (2D).

## *1.1. Motivação*

A RoboCup é uma excelente plataforma para desenvolvimento de novas ferramentas e técnicas para sistemas de controle autônomo em ambientes incerto e dinâmico. Em uma perspectiva educacional, é também um grande meio de ampliar os conhecimentos no projeto, construção, gerenciamento e manutenção de sistemas complexos.

O problema proposto pela liga de futebol simulado 2D, é que agentes joguem uma partida de futebol, seguindo regras e atuando em um ambiente dinâmico que adiciona complexidade do mundo real em suas simulações, visto que as percepções

são limitadas (RIBEIRO, 2010). Durante o jogo cada agente deve assumir uma função específica, atacando ou defendendo o grupo contra o time adversário. Estes e muitos outros problemas inerentes de um ambiente dinâmico são impostos aos agentes, que devem atuar e reagir prontamente.

Trabalhar com sistemas multiagentes em ambiente dinâmico onde há a necessidade de exprimir comportamento social de cada agente desenvolvendo um processo de construção da inteligência coletiva é o principal estímulo para este trabalho.

## *1.2. Objetivos do Trabalho*

O objetivo principal deste trabalho é o de desenvolver um time competitivo de Futebol de Robôs que atuará na liga de futebol simulado 2D da RoboCup.

Os objetivos específicos deste trabalho são:

- Estudo do simulador de futebol de robôs, as regras da partida e também do protocolo de comunicação entre agente e simulador;
- Realizar a simulação do futebol de robôs no ambiente virtual, possibilitando o estudo do comportamento do time e as técnicas de inteligência artificial;
- Desenvolver um time de futebol de robôs;
- Desenvolver componentes que controlem as estratégias e esquemas táticos que o time deve ter em cada momento do jogo.

Além disso, tem como objetivos secundários o grande potencial de pesquisa nas áreas de Inteligência Artificial, Sistemas Multiagente, Robótica, Comportamento Social dentre outras.

## *1.3. Metodologia*

Para o desenvolvimento deste trabalho inicialmente foi realizado um levantamento bibliográfico sobre arquitetura de agentes inteligentes, desenvolvimento de sistema multiagentes e do ambiente de simulação de futebol de robôs. Um estudo de táticas e esquemas táticos para basear o comportamento social do time também foi necessário além de uma pesquisa complementar para aprofundar os conhecimentos da linguagem Python. Através dessas leituras e com base nos conhecimentos adquiridos

foi modelado um sistema multiagente de um time de futebol simulado e então codificado baseado na linguagem estudada. Foram realizados testes nos componentes desenvolvidos que regem as estratégias de time e servem de base para aprimoramento posterior.

## *1.4. Organização da Monografia*

Esta monografia está organizada em cinco capítulos que tratam sobre os conhecimentos relacionados e desenvolvimento do trabalho. No capítulo 2 é feita uma revisão de literatura sobre as idéias que são discutidas a respeito de agentes e sistemas multiagente, ambientes dinâmicos aplicados ao Futebol de Robôs, bem como o funcionamento do ambiente de simulação. No capítulo 3 são expostos vários trabalhos que se encontram relacionados. O capítulo 4 é reservado ao relato do desenvolvimento do time UESBots, soluções, considerações e técnicas. No capítulo 5 discutem-se as conclusões do trabalho e possibilidades de trabalhos futuros.

# 2. Fundamentação Teórica

Neste capítulo são apresentados alguns conceitos teóricos relacionados ao trabalho descrevendo as terminologias básicas e metodologias.

## *2.1.Agentes*

No final da década de 70 e durante toda a década de 80, a comunidade científica começou a explorar novas áreas onde sistemas de IA pudessem ter um domínio mais dinâmico. Ao invés de olhar para resultados simulados, simbólicos em mundos artificiais, começaram a explorar as possibilidades de interações complexas com o mundo físico, através de um mecanismo denominado agentes.

Entretanto, uma barreira surgiu: muitos termos estavam sendo usados para descrever agentes, como: *intelligent agents*, *intelligent interfaces*, *adaptive interfaces*, *knowbots*, *softbots*, *userbots*, *taskbots*, *personal agents* e *network agents*, dentre outros (BRAGA et al. 2001). Sendo que cada termo utilizado referenciava o papel exercido pelo agente.

Os significados do termo “agente” vêm sendo discutidos por diversos pesquisadores da área que tentam responder perguntas como “o que é um agente?” ou “o que torna algo um agente?”. WOOLDRIDGE (1995 apud RIBEIRO, 2010) considera que a resposta a estas questões é tão imprecisa quanto uma resposta para a questão "o que é inteligência?" visto que, nos dois casos, no cenário computacional, não existe uma definição para o tema aceita universalmente.

Embora não exista uma definição genérica formada para agentes, esta se faz necessária para que o termo não acabe por perder seu significado. BRAGA (et al. 2001) define agente como sendo um sistema de computador que está situado em algum ambiente e que é capaz de executar ações autônomas de forma flexível neste ambiente, a fim de satisfazer seus objetivos de projeto. Existem três conceitos nesta definição: ambiente, autonomia e flexibilidade.

### *2.1.1. Propriedades*

Existem diversas propriedades encontradas em definições designadas por pesquisadores da área que visam agrupar os agentes em classes ou tipologias. Elas são apresentadas nos agentes de acordo com as funcionalidades desejadas para o seu uso.

WOOLDRIDGE (1995 apud RIBEIRO, 2010) relaciona as seguintes propriedades a serem consideras em um agente:

- **Autonomia**: os agentes podem operar sem a intervenção de humanos ou de outros agentes;
- **Sociabilidade**: capacidade de interagir através de troca de informações vindas do ambiente ou de outros agentes;
- **Reatividade**: os agentes são capazes de reagir a mudanças no ambiente;
- **Mobilidade**: capacidade de um agente de locomover-se entre diferentes ambientes;
- **Pró-atividade**, iniciativa: os agentes não são apenas entidades que reagem a um estímulo. Tem caráter empreendedor e podem atuar guiados por seus objetivos.

### *2.1.2. Tipos de Agentes*

Dentre as diversas classificações que se podem dar a agentes a mais comum é a mostrada por RUSSEL (et al. 2004) onde agentes são classificados nos tipos:

- **Agentes Reativos:** São geralmente agentes simples que possuem um mapeamento de situações e respostas associadas, ou seja, quando há alterações em um estado ambiental, o agente executa a ação correspondente para satisfazer o novo estado. Tem semelhanças com modelos de organização biológica ou de comportamento animal, tais como a sociedade das abelhas ou formigas. Existem ainda agentes conhecidos como reativos de estado interno ou agentes autômatos que diferentemente do agente reativo simples, utiliza tanto as percepções atuais quanto as passadas.
- **Agentes Cognitivos:** Este tipo de agente também pode ser chamado de simbólico ou deliberativo e se baseia em um modelo organizacional social humano. Este modelo possui uma memória que armazena um histórico das ações passadas e auxilia no planejamento de ações futuras, embora só seja alterado pelo agente em resposta a novas informações do ambiente. Com base na interpretação, o agente estima quais ações serão necessárias para atingir o objetivo proposto e também quais suas conseqüências e só posteriormente executa então as ações que possibilitarão a sua realização.

- **Agentes Híbridos**: Combinam as características das duas anteriores e tem capacidades deliberativas e reativas.
- **Agente Otimizador:** Permite escolher melhor compromisso entre vários objetivos conflitantes ou vários objetivos com probabilidades diferentes de serem alcançados. Tem como principal vantagem a utilização de ambiente sem restrição. Apresenta como desvantagens a falta de adaptabilidade assim como as abordagens existentes que tendem a ser pouco escaláveis.
- **Agente Deliberativo:** Este tipo de agente realiza uma associação indireta de percepção-ação, mediada pelo modelo atual do ambiente, com objetivo explícito, e pela previsão de estados futuros do ambiente resultante de uma seqüência de ações. Ele encadeia regras para construir um plano multi-passo necessário para atingir o objetivo a partir do modelo atual. Tem como principal vantagem o fato de tomar ações mais relevantes e seguras e como desvantagem o custo da deliberação, que pode ser excessivo em ambientes de tempo real.
- **Agente Adaptativo:** Tem como vantagem principal a adaptabilidade uma vez que possui a capacidade de aprender e de adaptar-se ao ambiente em que está inserido. Isso se deve à presença de um componente de análise crítica de desempenho associado a um componente de aprendizagem de conhecimento.

Para cada um dos tipos um modelo de arquitetura é definido, tornando o agente apropriado para seguir seus objetivos através das condições impostas pelo ambiente.

### *2.1.3. Arquitetura de Agentes*

Algumas das arquiteturas que os agentes podem ser classificados são:

#### *2.1.3.1. Arquitetura de Subsunção*

A arquitetura de subsunção foi uma das primeiras propostas para agentes. A idéia básica da arquitetura de subsunção é a decomposição do controle em camadas que correspondem a comportamentos elementares. Cada camada é um autômato de estado finito e existe uma hierarquia de prioridade entre eles.

A Figura 01 esquematiza a arquitetura de subsunção. Na figura, as setas de entrada nas camadas correspondem a informações obtidas através de sensores e as setas de saída correspondem a ações a serem executadas. A ação oriunda da camada superior tem prioridade sobre a ação oriunda da camada inferior.

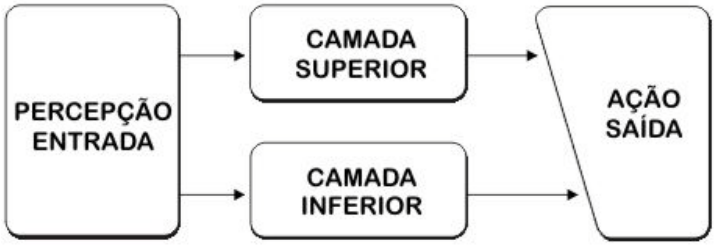


**Figura 01:** Arquitetura de Subsunção

#### *2.1.3.2. Arquitetura baseada em Lógica*

Neste tipo de arquitetura, o comportamento inteligente do sistema pode ser gerado a partir da representação simbólica do seu ambiente através de um conjunto de fórmulas lógicas. A proposta de construção de sistemas inteligentes sugere que um comportamento inteligente pode ser construído ao se prover uma representação simbólica do seu ambiente e do comportamento desejado para o mesmo, além de meios para manipular sintaticamente essa representação.

O processo de tomada de decisão de um agente é modelado a partir de um conjunto de regras de decisão. É necessário que o agente encapsule no seu código tanto as Regras quanto a Base de Conhecimento, de tal forma que, caso uma fórmula seja derivada a partir deles, possa ser concluído que existe um termo que descreve a melhor ação possível para dada situação (WEISS, 1999 apud BARBOSA, 2005).

Este método possui a desvantagem da complexidade computacional inerente da “prova de teoremas”, que torna questionável a utilização desse tipo de agentes em ambientes com restrições de tempo de resposta (WEISS, 1999 apud BARBOSA, 2005).

#### *2.1.3.3. Arquitetura baseada em Metas*

Um agente orientado a metas é pró-ativo, onde é prevista uma relação entre uma pré-condição e o efeito da pós-condição. Quando é feita uma chamada a um procedimento e uma pré-condição conduz a uma pós-condição verdadeira, a meta será

alcançada (WEISS, 1999 apud RIBEIRO, 2010). O conceito de meta pode apresentar dois aspectos:

Declarativo: onde é descrito o ambiente em que se deseja alcançar e é notável um raciocínio sobre propriedades fundamentais das metas. Este conceito assegura a habilidade de raciocínio sobre a meta, sem o qual não poderia haver verificação sobre a possibilidade de realizar a meta ou se há interferência entre elas.

Procedural: que traça um conjunto de planos para que a meta seja alcançada de maneira eficiente em ambiente bastante dinâmico e oferece ao agente a capacidade de produzir metas reais e possíveis de se alcançar.

Agentes orientados a metas são adequados para ambientes que não se alteram no período de execução do procedimento onde a meta é válida até que este finalize não sendo propícia para ambientes dinâmicos (WINIKOFF, 2002 apud BARBOSA, 2005).

#### *2.1.3.4. Arquitetura baseada em Camadas*

Esta abordagem, também conhecida como arquitetura híbrida, combina componentes da arquitetura reativa que pode reagir a eventos ocorridos no ambiente sem se ocupar com raciocínio complexo e da deliberativa que contem o modelo simbólico do mundo, desenvolve planos e toma decisões. Normalmente as ações do componente reativo podem ter precedência sobre o cognitivo, de modo a fornecer uma rápida resposta a algum evento importante do ambiente (WEISS, 1999 apud RIBEIRO, 2010). Na arquitetura em camadas os vários subsistemas são organizados em uma hierarquia de camadas interagindo por meio de processos lógicos que dão a cada camada a possibilidade de tomada de decisões sob diferentes níveis de abstração do ambiente (WEISS, 1999 apud RIBEIRO, 2010). De acordo com o fluxo de tomada de controle esta arquitetura pode ser classificada em camadas horizontais ou verticais.

- **Horizontal:** nas arquiteturas em camadas horizontais, como mostrado na Figura 02, cada camada de software está diretamente conectada ao sensor responsável pela entrada de percepções do mundo para formular suas próprias instruções de saída. Dessa forma, cada camada pode ser considerada um pequeno agente que produz sugestões sobre qual ação executar levando em conta o comportamento programado na camada. Dependendo da necessidade de diferentes comportamentos, mais camadas

podem ser acrescentadas ao agente. Para não haver uma competição entra as diferentes propostas de ação, uma camada com função de mediador deve ser inserida. Entretanto, o excesso de camadas pode provocar um gargalo na tomada de decisão do agente e reduzir seu desempenho global (WEISS, 1999 apud RIBEIRO, 2010).

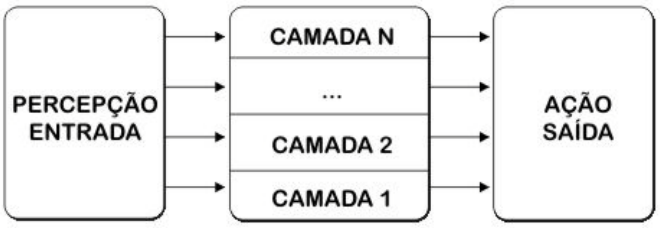


**Figura 02:** Arquitetura de Camadas Horizontais (adaptado de RIBEIRO, 2010)

- **Vertical:** nas arquiteturas em camadas verticais, exibida na Figura 03, o sensor de percepções fica conectado somente a uma camada de entrada e o fluxo de controle segue seqüencialmente pelas demais camadas. Esse fluxo pode ser classificado como de passagem única ou dupla. No fluxo de uma passagem cada camada faz sua interpretação das percepções e avalia sugestões vindas das camadas inferiores até que na camada final é definida a ação mais adequada para ser enviada aos atuadores do agente. Já no fluxo de duas passagens, as informações fluem para os níveis mais altos da organização, e quando alcançam o topo, o controle é novamente passado às camadas inferiores que refinam as possíveis ações para a mais adequada (WEISS, 1999 apud RIBEIRO, 2010).

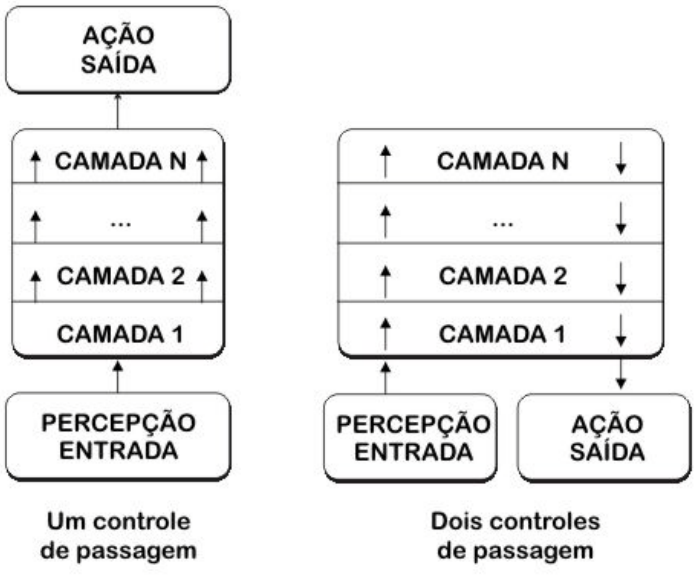


**Figura 03:** Arquitetura de Camadas Verticais (adaptado de RIBEIRO, 2010)

## *2.2. Sistemas Multiagentes*

Os SMA visam fornecer os princípios para a construção de sistemas complexos envolvendo múltiplos agentes. Tem como foco principal coordenar o comportamento inteligente de um conjunto de agentes autônomos, para obter a solução de um problema apresentado que está além da capacidade de resolução de cada um individualmente (OLIVEIRA, 1996 apud SOUZA, 1996). Um agente em um SMA tem sua existência própria, não dependente dos outros agentes do ambiente. No entanto, para alcançar seus objetivos ele eventualmente precisará da ajuda de outros.

### *2.2.1. Inteligência Artificial Distribuída*

A IAD tem como objetivo construir e desenvolver os sistemas compostos de entidades com graus de autonomia e inteligência variáveis e que cooperem entre si para o aumento de desempenho, visando resolver um problema global. Os SMA podem ser classificados a depender da resolução de problemas abordados:

- **Resolução Distribuída de Problemas (RDP):** Nesses sistemas a coordenação das atividades dos agentes é centralizada e são fortemente acoplados. Os múltiplos agentes têm o comportamento definido através da existência de regras já pré-estabelecidas. Inicialmente todos os agentes conhecem os demais agentes da comunidade e todos trabalham de maneira benevolente para atingirem um dado objetivo, existindo desta forma confiança mútua em relação às suas interações. O planejamento das ações desenvolvidas por cada agente é efetuado inicialmente pela divisão do problema em subproblemas que são entregues aos vários agentes do grupo.
- **Sistemas Multiagentes:** Os agentes podem ter tanto objetivos globais como também objetivos locais. A coordenação do comportamento inteligente dessa comunidade é descentralizada, de maneira que o resultado é obtido com a integração do conhecimento, capacidade, objetivos, e planos dos diferentes agentes autônomos. Os agentes participam mais autonomamente na decisão de realização das suas próprias ações. Não existe um planejamento inicial e os agentes podem ou não ter conhecimento prévio dos demais agentes.

### *2.2.2. Comunicação entre Agentes*

A comunicação fornece a base necessária para a realização da cooperação entre múltiplos agentes. Através da comunicação os agentes podem ter uma visão menos local do problema e adquirir o conhecimento e sincronia necessários, possibilitando resoluções mais rápidas. Porém, se a comunicação for excessiva será prejudicial, pois a carga de comunicação poderá ser maior que o trabalho efetivamente realizado (COSTA, 1999).

Atualmente diversos sistemas de computação envolvem múltiplos computadores interagindo de forma distribuída. Assim, a habilidade de comunicação é um importante atributo que os agentes inteligentes devem possuir. Ambientes que possuem mais de um agente praticamente exigem o intercâmbio de informações. Dessa forma, torna-se claro a necessidade de uma linguagem de comunicação comum e definir qual a linguagem ideal pode ser um problema. Atualmente, não existe uma linguagem padronizada e aceita mundialmente para a representação de informações trocadas por agentes. A presença de uma linguagem comum para troca de mensagens é uma das características que diferencia um agente de um objeto na programação orientada a objetos. De acordo com FINN (et al. 1993 apud COSTA, 1999), os níveis de comunicação entre agentes podem ser classificadas em:

- **Transporte**: como agentes enviam e recebem mensagens.
- **Linguagem**: qual o sentido de mensagens individuais.
- **Política**: como os agentes estruturam conversações.
- **Arquitetura**: como conectar sistemas utilizando com protocolos existentes.

## *2.3. Ambiente de Agentes*

Em diferentes ambientes os agentes devem tomar suas decisões baseados em percepções do mundo a sua volta considerando objetivos próprios e coletivos. Um ambiente provê as condições em que uma entidade (agente ou objeto) existe, definindo propriedades do mundo em que um agente realiza suas ações e afeta diretamente o seu desenvolvimento. Com relação às dimensões os agentes podem estar inseridos em um universo físico (robôs), de software ou de ambientes de simulação em três dimensões. Os ambientes podem ser classificados de acordo com suas características, segundo RUSSEL (et al. 2004 apud RIBEIRO, 2010) elas são:

- **Totalmente Observável x Parcialmente Observável:** se os sensores de um agente são capazes de captar todos os aspectos importantes para que ele possa realizar alguma ação em um ambiente ele é considerado completamente observável. Sua vantagem é que não há necessidade que o agente armazene suas percepções em estados internos para ter o controle do mundo em que atua. Um ambiente pode passar a ser considerado como parcialmente observável se as informações passadas pelos sensores apresentarem ruído ou se o agente tiver sua visão encoberta por qualquer objeto no ambiente.
- **Determinista x Estocástico:** um ambiente é considerado determinista se o estado atual do agente e as ações executadas por ele determinarão completamente o próximo estado do ambiente, caso contrário ele é considerado estocástico.
- **Episódico x Seqüencial:** quando a experiência de um agente é dividida em episódios atômicos contendo uma seqüência de percepções e ações, a qualidade da ação depende apenas do episódio mais recente. Em ambientes seqüenciais a decisão tomada irá afetar as futuras decisões, o que torna o agente mais complicado que o episódico, pois o obriga a ponderar nas conseqüências das ações.
- **Estacionário x Dinâmico:** se enquanto um agente estiver raciocinando, o ambiente em que ele se encontra estiver mudando, ele é considerado dinâmico. A grande vantagem do ambiente estático é a falta de preocupação em continuar a observar o mundo enquanto ele estiver pensando porque ele não irá mudar.
- **Discreto x Contínuo:** um ambiente é discreto se nele é fixado um número finito de ações e percepções, em RUSSEL (et al. 2004 apud RIBEIRO, 2010) um exemplo de ambiente discreto dado é o jogo de xadrez, e para um ambiente contínuo um motorista de táxi imerso em uma cidade.
- **Agente Unitário x Multiagente:** ambientes onde um agente tenta resolver um problema como um quebra-cabeça são caracterizados como agente unitário. Em ambientes multiagente existem agentes trabalhando para resolver um problema proposto, sendo que podem estar competindo ou cooperando uns com os outros.

### *2.3.1. Ambientes de Simulação*

Um ambiente de simulação é uma plataforma de desenvolvimento de aplicações e realização de pesquisas em Inteligência Artificial. Estes após serem inicializados, começam seu ciclo de processamento, onde serão geradas as percepções a partir do estado atual do ambiente, e então estas percepções são enviadas aos agentes. O simulador também tem como função receber e processar as ações escolhidas pelos agentes, para que seja possível a atualização do estado do ambiente (RIBEIRO, 2010).

A RoboCup propõe o uso de um jogo de futebol como uma plataforma para uma ampla variedade de pesquisas em IA e robótica. O ambiente formado pelos agentes e o simulador é bastante dinâmico, pois enquanto um agente está inferindo em qual ação ele deve tomar, outros podem estar realizando alguma ação, a bola pode estar em movimento e o ambiente, de uma forma geral, sofre influência de todas as ações.

## *2.4. Futebol de Robôs*

A idéia de robôs jogarem futebol foi mencionada pela primeira vez pelo professor Alan Mackworth (University of British Columbia, Canadá) em um artigo intitulado "On Seeing Robots" apresentado no VI-92, em 1992 (ROBOCUP, 2011).

> *RoboCup é uma iniciativa internacional de pesquisa e educação. É uma tentativa de promover a Inteligência Artificial - IA e a investigação em robótica inteligente fornecendo um problema padrão onde uma ampla diversidade de tecnologias possam ser integradas e estudadas, bem como ser usada como parte em projetos de orientação educacional (ROBOCUP, 2011).*

O futebol de robôs inclui ligas que se dividem em ligas de robôs reais e liga de simulação. As competições entre robôs autônomos funcionam como laboratório de capacitação de robôs para a realização autônoma de tarefas e para a formação de sociedades artificiais, visando a realização de tarefas cooperativas.

Antes do futebol de robôs, os pesquisadores de IA já haviam adotado outro desafio, investigado nas últimas quatro décadas: construir um programa/computador capaz de vencer o campeão mundial de xadrez utilizando as regras oficiais da Federação Internacional de Xadrez. Em 1997 ocorreu um marco importante para a área de IA quando o computador desenvolvido pela IBM e apelidado de Deep Blue

(DEEP BLUE, 1997) derrotou o campeão mundial de xadrez humano, o russo Gary Kasparov. Após esse marco, o futebol de robôs se tornou o desafio padrão.

A principal diferença entre os problemas do xadrez e do futebol de robôs é que o segundo consiste em um controle distribuído, onde vários agentes tem de agir autonomamente e coordenar-se a fim de cooperar para atingir um objetivo comum. O problema se torna crítico, uma vez que o ambiente é dinâmico e a mudança de estado é em tempo real. As diferenças principais entre o domínio do Xadrez e o RoboCup podem ser visualizadas na Tabela 01.

**Tabela 01:** Comparação entre Xadrez e Futebol de Robôs (ROBOCUP, 2011b)

| | **Xadrez** | **RoboCup** |
|---|---|---|
| **Ambiente** | Estático | Dinâmico |
| **Mudança de Estado** | Seqüencial | Tempo Real |
| **Acesso à informação** | Completa | Incompleta |
| **Leituras de Sensor** | Simbólica | Não Simbólica |
| **Controle** | Centralizado | Distribuído |

Quando a internacionalização da idéia do Futebol de Robôs emergiu, não haviam regras a fim de garantir compatibilidade das equipes. Para suprir essa lacuna, pesquisadores sul-coreanos fundaram a FIRA (Federation of International Robot-soccer Association) em 1997. Paralelamente à iniciativa da FIRA, a empresa japonesa Sony incentivou o surgimento de competições de Futebol de Robôs em escolas e universidades, o que resultou na criação de outra federação denominada RoboCup.

## *2.5. Robocup*

As atividades atuais da RoboCup consistem em (ROBOCUP, 2011):

- Conferências Técnicas;
- Conferências e Competições Internacionais da RoboCup;
- Programas de Desafios RoboCup;
- Programas de Educação;
- Desenvolvimento de infra-estrutura.

A RoboCup organiza a Copa do Mundo de Futebol de Robôs e conferências, onde os pesquisadores podem avaliar o progresso na área.

### *2.5.1. Categorias*

Atualmente, RoboCup tem quatro domínios principais (ROBOCUP, 2011):

#### *2.5.1.1. RoboCupSoccer*

O futebol, além de ser um esporte bastante popular é também um jogo com problemas individuais e coletivos. Essa liga é subdividida em categorias com competições com agentes físicos e agentes simulados (ROBOCUP, 2011).

- **Middle Size League:** Jogada por duas equipes de cinco robôs com uma altura que varia entre os 50 e os 90cm e com comunicação wireless. Não é permitida qualquer intervenção humana durante o jogo exceto nas reposições de bola. (ROBOCUP, 2011).
- **Small Size League:** Nesta liga os robôs são bem menores. Embora sejam independentes uns dos outros, todo o processamento é efetuado num computador central, baseado em informação vinda de uma câmera de vídeo colocada por cima do campo de jogo (ROBOCUP, 2011).
- **Four-legged league:** Como o nome da liga indica, esta é jogada por robôs de quatro patas, o cão robótico AIBO da Sony formando equipes de cinco jogadores (ROBOCUP, 2011).
- **Simulation league:** Nesta liga não existem robôs/agentes físicos e os agentes são simulados. Um simulador fornece aos agentes todos os dados que seriam obtidos na realidade através dos seus sensores. Cada agente é um processo separado (sendo 11 de cada equipe). Existe simulação em duas ou três dimensões (ROBOCUP, 2011).
- **Humanoid league:** Esta competição baseia-se ainda em aspectos básicos dos jogadores robôs, não existindo atualmente jogos de futebol como nas outras ligas (ROBOCUP, 2011).
- **E-league:** Esta é a mais recente liga do Robocup e centra-se nos problemas de alto-nível, usando plataformas simples e tecnologias generalizadas quanto à visão e comunicação (ROBOCUP, 2011).

#### *2.5.1.2. RobocupRescue*

A RobocupRescue consiste na utilização de robôs para busca e salvamento em cenários de catástrofes artificiais e é subdividido em agentes físicos e simulados.

#### *2.5.1.3. RobocupJunior*

Esta categoria visa iniciar jovens estudantes do ensino primário e secundário nas áreas de robótica e inteligência artificial (ROBOCUP, 2011).

#### *2.5.1.4. Robocup@Home*

Tem seu foco principal nas aplicações em mundo real e na interação homem-máquina com robôs autônomos. O objetivo é promover o desenvolvimento de robôs que irão auxiliar os humanos no seu dia-a-dia. Os cenários envolvem ambientes de uma casa. (ROBOCUP, 2011).

### *2.5.2. Futebol Simulado 2D*

O Futebol Simulado 2D é uma simulação do jogo de futebol, apresentando várias das características presentes no mundo real. Com exceção da terceira dimensão, todas as outras características utilizadas representam bem um jogo de futebol real. A utilização de variáveis aleatórias torna a simulação realista para os usuários, contribuindo consideravelmente para as pesquisas.

A simulação é composta por três sistemas distintos: o servidor, o monitor e os clientes (jogadores e técnicos), conforme ilustrado na Figura 04. O funcionamento da simulação em alto nível é bastante simples, a cada ciclo de simulação os passos descritos a seguir são repetidos até que o tempo total de jogo, tipicamente 6.000 ciclos, seja atingido:

- **Passo um:** O servidor envia informação para os clientes e o monitor. Esta informação é referente ao que está “acontecendo” no ciclo. Algumas partes desta informação seriam: velocidade de objetos (jogadores, bola, etc), posição de objetos, condições do próprio jogador, resumindo, o que o jogador estaria vendo, sentindo e ouvindo.

- **Passo dois:** Quando o monitor recebe esta informação, ele representa tudo em uma forma visual para que a equipe e os espectadores possam acompanhar o jogo.
- **Passo três:** Quando o cliente recebe esta informação, ele analisa o que está acontecendo segundo um algoritmo pré-estabelecido pelos responsáveis pelo time, e toma a decisão determinada pelo algoritmo (correr, virar, chutar, etc) e envia o comando correspondente a essa decisão para o servidor.
- **Passo quatro:** O servidor então calcula as novas condições, baseado nas condições anteriores e nos comandos enviados pelos clientes e realiza novamente o Passo um.

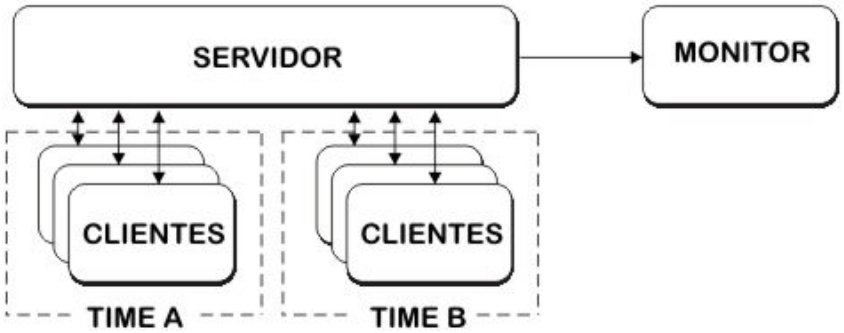


**Figura 04:** Componentes da Simulação de Futebol 2D.

A tarefa de um time que se propõe participar de uma partida nesta categoria é conceber e implementar o algoritmo responsável pelo Passo 3. Tal algoritmo deve ser capaz de processar as informações recebidas do servidor de modo a tomar a melhor decisão para marcar gols ao mesmo tempo em que evita a marcação do adversário.

### *2.5.3. O Servidor RoboCup*

O servidor RoboCup é um sistema que permite múltiplos agentes competirem em um jogo de futebol através da simulação de todos os movimentos da bola e jogadores em um campo virtual. É baseado no estilo cliente-servidor, e não impõe qualquer restrição na arquitetura dos times, basta que a comunicação entre cliente-servidor tenha suporte ao User Datagram Protocol (UDP) e obedeça aos protocolos de comunicação utilizados (CHEN et al., 2003 apud RIBEIRO, 2010).

As regras de formação das equipes seguem as regras do futebol, onde até doze agentes compõem cada time, sendo que um é o treinador (coach). Cada cliente é um

processo distinto e controla os movimentos de um agente. Os jogadores podem enviar comandos para o servidor requisitando ações que eles desejam tomar (chutar a bola, virar, correr, etc.). O servidor recebe essas mensagens, interpreta as requisições, atualiza o ambiente de acordo e depois retorna percepções sensoriais. A comunicação entre os agentes pode ser feita somente através de informações auditivas com os comandos SAY (falar) e HEAR (ouvir). Em outras palavras, os clientes que representam os jogadores não podem trocar informações diretamente (CHEN et al., 2003 apud RIBEIRO, 2010).

É importante mencionar que o servidor é um Sistema de Tempo Real (STR) que trabalha com intervalos discretos (ciclos). Cada ciclo tem uma duração específica e ações precisam ser efetuadas em um dado ciclo devem chegar até o servidor no intervalo certo. Caso contrário, o jogador perderá oportunidades de jogadas e causará uma baixa no desempenho para todo o time (CHEN et al., 2003 apud RIBEIRO, 2010).

#### *2.5.3.1. Percepções sensoriais*

As percepções sensoriais são divididas em informação visual, auditiva e corporal, cada uma tem o seu respectivo formato de mensagem.

#### *2.5.3.2. Informação Visual*

O campo virtual utilizado pelo servidor possui as dimensões de 105x68m e contem objetos estáticos que servem para ajudar o jogador a se localizar no campo. As informações visuais desses objetos são enviadas para os jogadores a cada 150ms e chegam no formato básico *(see Time "ObjName" Distance Direction DistChange DirChange BodyDir HeadDir)*, onde:

**Tabela 02:** Formato de mensagem visual do Servidor RoboCup

| | | |
|---|---|---|
| *ObjName* | Outros jogadores no formato "(p TeamName UniformNumber [goalie])" ou objetos como bandeiras e linhas do campo. | |
| | *TeamName* | Nome da equipe em formato de texto |
| | *UniformNumber* | Número do uniforme que varia de 1 a 11 |
| | *[goalie]* | Aparece quando o jogador é um goleiro |
| *Time* | Ciclo atual na simulação do servidor | |
| *Distance* | Distância do jogador até o objeto observado | |

| | |
|---|---|
| *Direction* | Ângulo formado com o jogador varia de -180 a 180 graus |
| *DistChange*[1] | Mudança de distância |
| *DirChange*[1] | Mudança de direção |
| *HeadFaceDir* | Direção da cabeça do jogador -180 a180 graus |
| *BodyFaceDir* | Direção do corpo -180 a 180 graus |

1. *DistChange e DirChange representam a mudança da distância e direção desde a ultima visão do objeto, se o objeto é algo imóvel como gol, bandeiras ou linhas, esses valores serão omitidos (CHEN et al., 2003 apud RIBEIRO, 2010).*

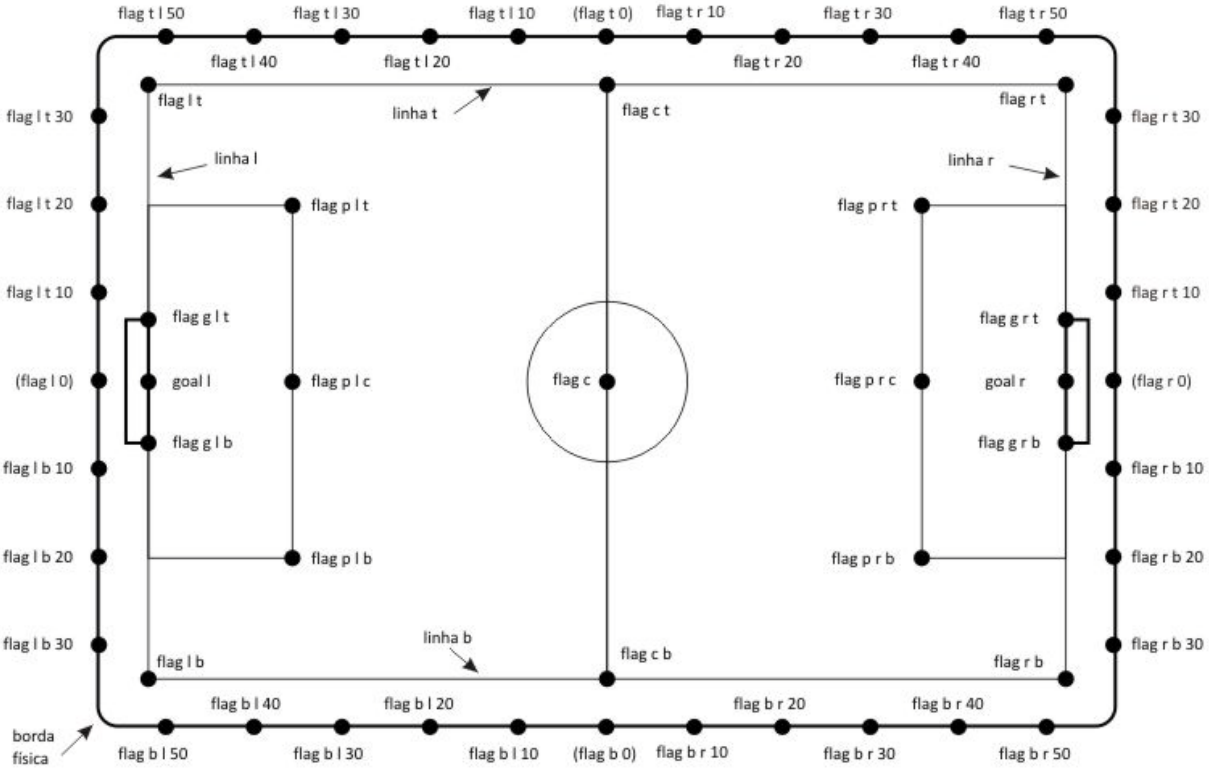


**Figura 05:** Objetos delimitadores do campo virtual (adaptado de RIBEIRO, 2010).

**Tabela 03:** Objetos delimitadores do campo virtual

| | |
|---|---|
| *flag c* | bandeira que serve para representar o centro do campo, como indicado pela letra "c" |
| *linha l\|r\|t\|b* | linhas laterais e de fundo de cada gol |
| *goal l\|r* | gols de cada lado, esquerdo e direito respectivamente |
| *flag r\|l t\|b* | equivalem às bandeiras de escanteio de um campo real |
| *flag p l\|r t\|b* | linhas que demarcam a área de pênaltis (grande área) |

| | |
|---|---|
| *flag g l\|r t\|b* | traves do gol de cada lado do campo |
| *flag t\|b l\|r #* | outras bandeiras estão localizadas a 5 metros fora do campo, como por exemplo, a bandeira “flag t l 30” que está localizada a 5 metros da linha superior (line t) e a 30 metros da linha central |

No campo virtual do servidor, visto na Figura 05, existem objetos posicionados em diversos locais e são utilizados como referencia para que o agente possa se localizar no campo. Esses objetos são classificados como bandeiras, linhas e gol e são explicados na Tabela 03.

A informação visual enviada para os agentes corresponde aos objetos identificados pelo robô, no setor visível do jogador. Esse setor visível pode assumir três diferentes ângulos: normal [-45, 45], amplo [-90, 90], direcionado [-22.5, 22.5]. Cada um dos ângulos de visão pode assumir o valor alto ou baixo para a qualidade da informação visual (RIBEIRO, 2010). A Figura 06 explica a visão do agente.

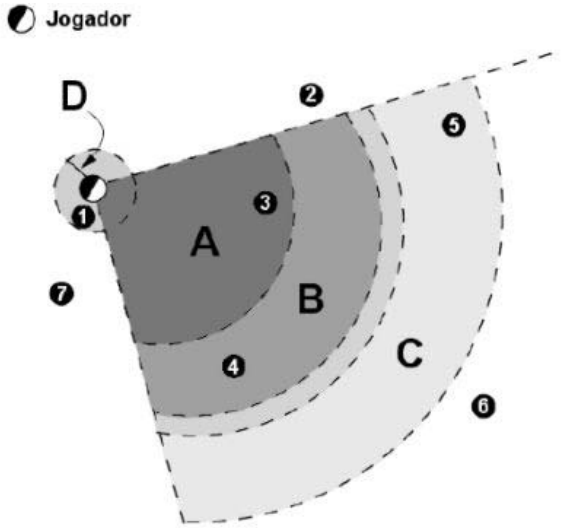


**Figura 06:** Cone de visão do agente (CHEN et al., 2003 apud RIBEIRO, 2010)

Na Figura 06 os objetos correspondentes aos números 2 e 7 são os únicos fora da visão do jogador, os demais se encontram em situações de exposição diferentes e, a depender do quão longe estão, a precisão na identificação é afetada. De acordo com Chen et al. (2003 apud RIBEIRO, 2010) os valores de distancias são:

- **Área A:** Corresponde a 20 metros de distância do jogador, dentro desse distancia é possível identificar tanto o número do uniforme como o nome da equipe de outro jogador;

- **Área B:** Corresponde a 40 metros de distância do jogador, até essa distância é possível identificar o número do uniforme de outro jogador com uma probabilidade de 50%, acima desse valor o número do uniforme não é visível;
- **Interseção entre as áreas B e C:** Ainda é possível identificar o nome da equipe de um jogador;
- **Área C:** corresponde a 60 metros de distância do jogador, até essa distância é possível identificar o nome da equipe de outro jogador com uma probabilidade de 50%, acima desse valor o nome da equipe não é visível;
- **Área D:** Corresponde a um raio de 3 metros do centro do jogador, tal que quando o objeto nela se encontra, é identificado mesmo estando fora do setor visível.

#### *2.5.3.3. Informação Auditiva*

Alem da informação visual, o agente recebe do servidor, mensagens enviadas pelo juiz informando alterações no estado do jogo (ocorrência de um gol, por exemplo) que não tem limite de distancia e mensagens enviadas por outros jogadores em até 50 metros de raio. O jogador poderá processar apenas uma mensagem de cada time a cada dois ciclos de simulação. Essas mensagens são ordenadas de acordo com a ordem de chegada. Existe ainda uma limitação quanto ao comprimento da mensagem, que não poderá ultrapassar 512 caracteres (CORTEM et al., 1999 apud RIBEIRO, 2010). A mensagem auditiva tem o formato básico *(hear Time Sender "Message")* que é explicado na Tabela 04.

**Tabela 04:** Formato de mensagem auditiva do Servidor RoboCup

| | |
|---|---|
| *Time* | Ciclo atual na simulação do servidor |
| *Sender* | Pode ser uma direção quando a mensagem foi originada de outro jogador ou então:<br>▪ *self* quando é do próprio jogador;<br>▪ *referee* quando vem do juiz do jogo;<br>▪ *online_coach_left*\|*right* quando vem de um dos técnicos dos times |
| *Message* | Mensagem recebida no formato de texto |

Um dos objetivos do Servidor RoboCup é a evolução dos Sistemas Multiagentes, e a eficiência da comunicação entre os agentes é um dos critérios principais. Os usuários devem realizar o controle dos múltiplos agentes com restrição na comunicação (CHEN et al., 2003 apud RIBEIRO, 2010).

#### *2.5.3.4. Informação Corporal*

O sensor de corpo reporta o atual estado físico do corpo de um jogador. Essas informações são enviadas automaticamente a cada 100 milissegundos (ms) e são úteis para o agente tomar decisões de acordo com os dados relativos ao seu corpo ou determinar se um comando foi aceito ou não pelo servidor (CHEN et al., 2003 apud RIBEIRO, 2010). A mensagem corporal tem o formato básico:

*(**sense**_body Time*
*(**view_mode** ViewQuality ViewWidth)*
*(**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))*

**Tabela 05:** Formato de mensagem corporal do Servidor RoboCup

| | |
|---|---|
| Time | Ciclo atual na simulação do servidor |
| ViewQuality | Corresponde a qualidade de visão e pode te valores *high* ou *low* |
| ViewWidth | Corresponde à abertura do ângulo de visão e podem ser *narrow*, *normal*, ou *wide* |
| Stamina | Representa a resistência do jogador, varia de 0 a 4000 por padrão |

| | |
|---|---|
| Effort | Representa o esforço feito pelo jogador |
| AmountOfSpeed | É o total de velocidade do jogador |
| DirectionOfSpeed | É a direção para onde o se movimenta, varia entre -180 a 180 graus |
| HeadAngle | A direção relativa da cabeça do jogador e varia entre -180 a 180 graus |

#### *2.5.3.5. Estados e Regras da partida*

O juiz da partida é o responsável por atribuir estados e regras de controle de acordo com as situações que o jogo se encontra, essas atribuições serão interpretadas pelos agentes e utilizadas para selecionar as estratégias de jogo que sejam mais adequadas (ROBOCUP, 2010). Os possíveis estados da partida são mostradas na Tabela 06.

**Tabela 06:** Estados de jogo de uma partida de futebol simulado

| | |
|---|---|
| before_kick_off | A partida encontra-se parada aguardando seu início. Este estado acontece no início do jogo (a partida se inicia ao comando kick_off do servidor ser acionado) ou durante os cinco segundos de intervalo dados pelo juiz após a ocorrência de um gol |
| time_over | Fim de jogo |
| play_on | Quando a bola está em jogo |
| kick_off_l<br>kick_off_r | Saída de bola para o time da esquerda e da direita |
| kick_in_l<br>kick_in_r | Reposição de bola em jogo para o time da esquerda ou para o da direita |
| free_kick_l<br>free_kick_r | Chute livre para o time da esquerda ou para o da direita |
| corner_kick_l<br>corner_kick_r | Cobrança de escanteio para o time da esquerda ou para o da direita |
| goal_kick_l<br>goal_kick_r | Cobrança de tiro de meta para o time da esquerda e da direita |
| goal_l<br>goal_r | Ocorrência de um gol em favor do time da esquerda ou para da direita |

| | |
|---|---|
| offside_l<br>offside_r | Impedimento. Todos os jogadores atacantes são posicionados num local em que não estejam fora de jogo. São considerados atacantes os jogadores que estão fora de jogo e os que estão a menos de 9.15 metros da bola |

Alem dos estados da partida os agente jogadores devem conhecer algumas regras de controle que segundo Chen (et al. 2003 apud RIBEIRO, 2010) são descritas como:

- **Gol:** Quando a bola se encontra dentro do gol, o juiz difunde para todos os agentes uma mensagem informando o acontecimento de um gol. Além disso, atualiza o placar, suspende a execução da partida por cinco segundos, move a bola para o centro do campo e muda o estado do jogo para kick_off, reiniciando a partida. Durante os cinco segundos de interrupção da partida todos os agentes devem voltar para seus lados do campo.
- **Saída de Bola:** Durante a saída de bola, inicio ou reinicio do jogo, todos os jogadores (agentes) devem estar em seu campo defensivo. Caso um jogador se encontre no campo adversário, este será movido para uma posição aleatoriamente escolhida em seu campo defensivo.
- **Bola Fora de Jogo:** Quando a bola se encontra fora das dimensões do campo, o juiz move a bola para a posição apropriada (e.g. lateral do campo, marca de escanteio ou pequena área) e muda o estado do jogo para kick_in (reposição de bola), corner_kick (escanteio) ou goal_kick (tiro de meta).
- **Desobstrução:** Após o goleiro segurar a bola, ou quando a partida se encontra em um dos seguintes modos, kick_off, corner_kick, goal_kick ou offside, o juiz remove qualquer jogador adversário raio de 9.15 metros da bola para o perímetro do circulo de mesmo raio centrado na bola.
- **Controle do modo de jogo:** Quando o jogo se encontra em um dos seguintes estados kick_off, corner_kick ou gool_kick, o juiz muda o estado do jogo para play_on após a bola ter sido colocada em jogo, por um dos jogadores do time que detém a posse da bola.
- **Final de Primeiro Tempo e Fim de Jogo:** O juiz suspende o jogo quando o relógio do soccerserver atingir 3000 ciclos de simulação (5 minutos) decretando o final do primeiro tempo. Difundirá então uma mensagem

informando o novo status do jogo. De forma similar, o juiz terminará o jogo ao atingir 6000 ciclos de simulação.

#### *2.5.3.6. Comandos executados pelos agentes*

Em resposta as informações visuais e auditivas, recebidas pelo agente, devem ser enviadas ao Soccer Server comandos que serão responsáveis pela ação do jogador. O Soccer Server aceita um novo comando a cada 20ms, entretanto existem limitações para utilização destes comandos (CHEN et al., 2003). De acordo com Chen et al. (2003 apud RIBEIRO, 2010) os comandos disponíveis e suas respectivas limitações são descritos na Tabela 07.

**Tabela 07:** Comandos que podem ser executados pelo agente no Servidor RoboCup

| | |
|---|---|
| *(turn moment)* | Permite que o jogador execute um giro de [-180, 180], graus em torno de seu próprio eixo. A este comando é associado um argumento *moment* (momento de inércia) que considera a inércia do objeto |
| *(turn_neck angulo)* | Permite que o jogador gire [-180, 180] graus em torno de seu próprio eixo, a parte superior do robô onde se encontra a câmera |
| *(dash potência)* | Incrementa a velocidade do jogador na direção atual com a potência especificada no argumento. A potência pode assumir um valor inteiro no intervalo [-100, 100] |
| *(kick potência direção)* | Chuta a bola com a potência de -100 a 100 na direção -180 a 180 graus. Este comando só terá efeito se a bola estiver na margem de chute especificada pelo Servidor |
| *(move X Y)* | Movimenta o jogador para a posição determinada. Os valores de X variam de -52.5 a 52.5 e os de Y variam de -34 a 34. Este comando é somente utilizado nos modos de partida *before_kick_off*, *goal_r* e *goal_l* |
| *(catch direção)* | Tenta agarrar a bola na direção especificada. A possibilidade de sucesso é definida no parâmetro do Servidor *catch_possibility*. A bola deve se encontrar em uma área definida como *goalie_cacthable_area* um retângulo de 2x1 metros. Esse comando somente poderá ser utilizado pelo goleiro |
| *(say mensagem)* | Difunde uma mensagem para todos os jogadores num raio de 50 metros |
| *(change_view ângulo* | Modifica o Angulo do setor visível para a *normal* [-45, |

| | |
|---|---|
| *qualidade)* | 45], *narrow* [22.5, 22.5], *wide* [-90, -90] e a qualidade da informação visual para *low* ou *high* |
| *(sense_body)* | O servidor retorna valores do estado físico atual do jogador, como os valores do modo de visão e o vigor do jogador |
| *(score)* | O servidor retorna a pontuação dos dois times |

Também existem comandos que o agente utiliza para iniciar a conexão com o servidor, isto é, avisar qual o nome do time e quando entrou em campo. Esses comandos são descritos na Tabela 08.

**Tabela 08:** Comandos de inicialização de agentes no Servidor RoboCup

| | |
|---|---|
| *(init nomeTime [(versãoSimulador)] [(goalie)])* | O primeiro comando que deve ser enviado para iniciar a conexão com o simulador, é necessário informar o nome do time e caso o jogador seja o goleiro |
| *(reconnect nomeTime númeroJogador)* | Este comando é útil para mudar o cliente em campo para o determinando numero do jogador |
| *(bye)* | Esse comando faz com que o simulador retire o jogador de campo e não mais aceite comandos do mesmo |

Chen (et al. 2003 apud RIBEIRO, 2010) afirma que o Servidor RoboCup aceita um novo comando a cada 20ms, entretanto apenas um dos comandos *turn, turn_neck, kick, move, dash, catch,* é executado a cada ciclo de simulação, que é de 100ms.

# 3. Trabalhos Relacionados

Onde foram analisados alguns trabalhos que tem relação e papel semelhantes ao proposto neste projeto.

## *3.1. Técnicas para construção de um time de futebol de robôs*

No trabalho de RIOS (2006) foi realizada uma avaliação das tecnologias disponíveis, técnicas de Engenharia de Software e Inteligência Artificial utilizadas para o desenvolvimento de times competitivos de Futebol de Robôs. Como trabalho teórico serviu para aumentar o entendimento do problema e também de norte para encontrar as soluções mais acertadas no desenvolvimento do time de Futebol de Robôs da UESB.

## *3.2. Protótipo de um time de futebol de robôs da UESB*

O trabalho de RIBEIRO (2010) mostra uma solução de implementação de agentes com componentes em camadas verticais e serviu de base para este trabalho que procurou estende-lo modificando o modelo e classes implementadas para que satisfizessem as necessidades apresentadas no desenvolvimento da camada de raciocínio do agente, um dos objetivos deste trabalho.

RIBEIRO (2010) propõe o desenvolvimento do agente utilizando a notação de Desenvolvimento Baseado em Componentes e a modelagem de um sistema básico, onde o agente tem capacidade para entender e atuar em um ambiente de forma coerente com as suas possibilidades. Foi realizada a implementação do protótipo constituído por todos os métodos das interfaces dos componentes e classes auxiliares. Uma inteligência reativa simples foi realizada para fim de testes. Após vários testes realizados foi constatado que o jogador é capaz de realizar algumas funções básicas, enviando e recebendo informações do servidor RoboCup com sucesso. Contudo concluiu que ao executar testes com agentes de outros times ficou clara a necessidade de uma inteligência mais apurada.

## *3.3. UvA Trilearn 2001*

O time UvA Trilearn foi desenvolvido na Faculdade de Ciência da Universidade de Amsterdam. Foi tema da dissertação de mestrado de BOER (et al. 2002), na qual propunha o desenvolvimento de um time com uma arquitetura dividida

em três camadas e serviu como base para comparação para este trabalho por possuírem arquitetura semelhante.

Encontram-se disponíveis para utilização somente dois modos simples de ações no código no time UvA Trilearn: O modo de manuseio da bola e o modo de posicionamento. Se o jogador estiver perto da bola, o modo manuseio da bola é executado. De outra forma, haverá então a ação de interceptar a bola. A ação de interceptar a bola consiste em posicionar o jogador no ponto de intersecção da trajetória da bola em relação à posição do adversário (BOER et. al, 2002).

O time UvA Trilearn, assim como este trabalho desenvolvido, apresenta como uma arquitetura de três camadas que realizam execução paralela dos dados, sincronização do ambiente, localização eficaz de objetos, estimação de velocidades usando filtros de partículas e uma camada hierárquica de competências. Estas semelhanças tornam este time um candidato ideal para comparações na construção do time, objeto deste trabalho.

# 4. O Time UESBots

Este capítulo será iniciado com uma breve contextualização sobre as características de desenvolvimento do sistema multiagentes, em seguida, serão apresentados as modelagens de agentes, modelo do mundo onde o agente está imerso, funcionamento dos sensores e atuadores. Serão mostradas também as rotinas desenvolvidas para o mecanismo de inteligência que correspondem às habilidades individuais de um jogador e às estratégias de time, como formações, posicionamentos e comunicação.

## *4.1. Contextualização*

Tomando-se como base a modelagem do protótipo para time de agentes jogadores com o objetivo de participar da liga de futebol 2D desenvolvida por RIBEIRO (2010), este trabalho tem como objetivos revisar e ampliar os mecanismos responsáveis pela tradução dos dados e manutenção do modelo de mundo.

A arquitetura de agentes inteligentes para a construção deste projeto é híbrida, baseada em camadas reativas e cognitivas. Problemas individuais do agente, que chamamos de habilidades individuais (chute a gol, interceptação da bola, etc.) e que exigem respostas rápidas são tratados pela camada reativa. Na camada cognitiva são tratadas as habilidades sociais ou coletivas, como estratégia de time, passes, dribles e posicionamento estratégico em campo.

A metodologia do trabalho de RIBEIRO (2010) está mantida e é baseada em componentes, pois tem se mostrado mais flexível para manutenção de código, organização de objetos e classes, além de se mostrar mais didático para estudos posteriores. Além de empregar métodos e técnicas que são boas práticas de engenharia de software.

A linguagem utilizada para a implementação deste trabalho foi a Python, linguagem de alto nível que dá suporte a comunicação por protocolo UDP, requisito básico para comunicação com o servidor RoboCup (ROBOCUP, 2010) . Além de também suportar *multi-threads*[1], essencial para o funcionamento interno dos agentes, e *multiprocessing*[2], essencial para inicialização do conjunto de agentes do time.

[1] *Processamento paralelo de partes do sistema. Basicamente significa um mesmo sistema processando paralelamente partes de código.*
[2] *Execução de múltiplos sistemas em processos separados. Basicamente significa quem há um controle da execução de vários sistemas (processos) em um sistema maior, conjunto destes processos.*

## *4.2. Arquitetura dos Agentes*

A arquitetura dos agentes utilizada no trabalho é divida em três camadas verticais conforme apresentado na Figura 07.

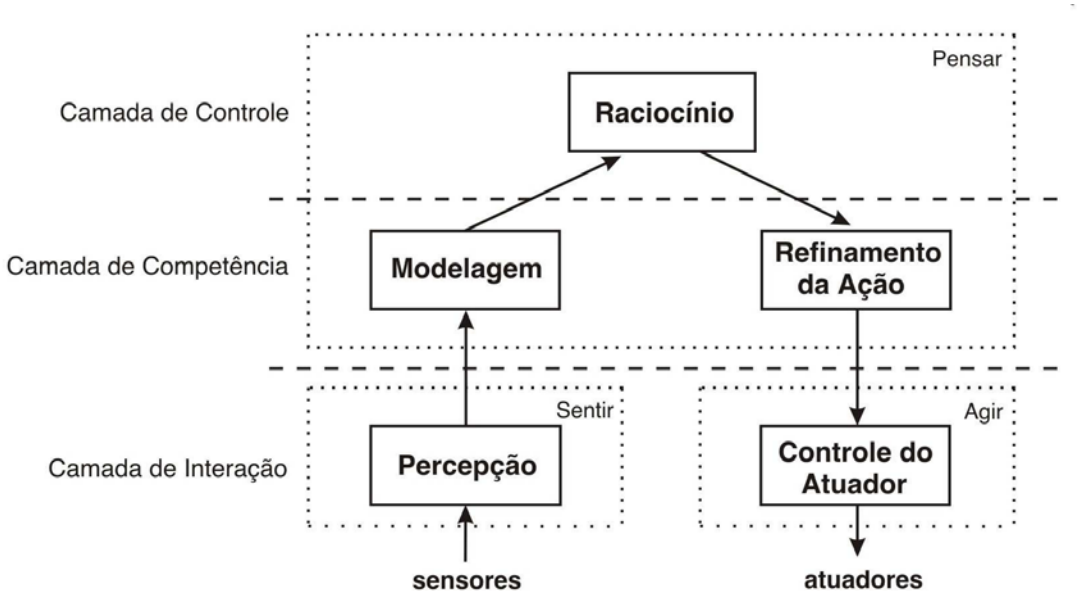


**Figura 07:** Arquitetura de Camadas dos Agentes Jogadores

As camadas têm as seguintes funcionalidades:

- **Camada de Interação:** responsável pela interação com o servidor. Recebe as informações dos sensores e envia para camada de competência. Envia para o servidor as ações recebidas pela camada de competência, de acordo a respectiva estratégia;
- **Camada de Competência:** utiliza as funcionalidades enviadas pela camada de interação a fim de construir um modelo abstrato do jogo e para programar os diversos perfis de cada agente;
- **Camada de Controle:** responsável pelo raciocínio do sistema. A melhor possibilidade de ação é selecionada da camada de competência, dependendo do estado atual do jogo e da estratégia atual do time.

## *4.3. Modelo das Camadas*

Na Figura 08 é apresentado o diagrama UML de classes, no qual é possível identificar as relações de associação, composição e generalização entre as três camadas da arquitetura.

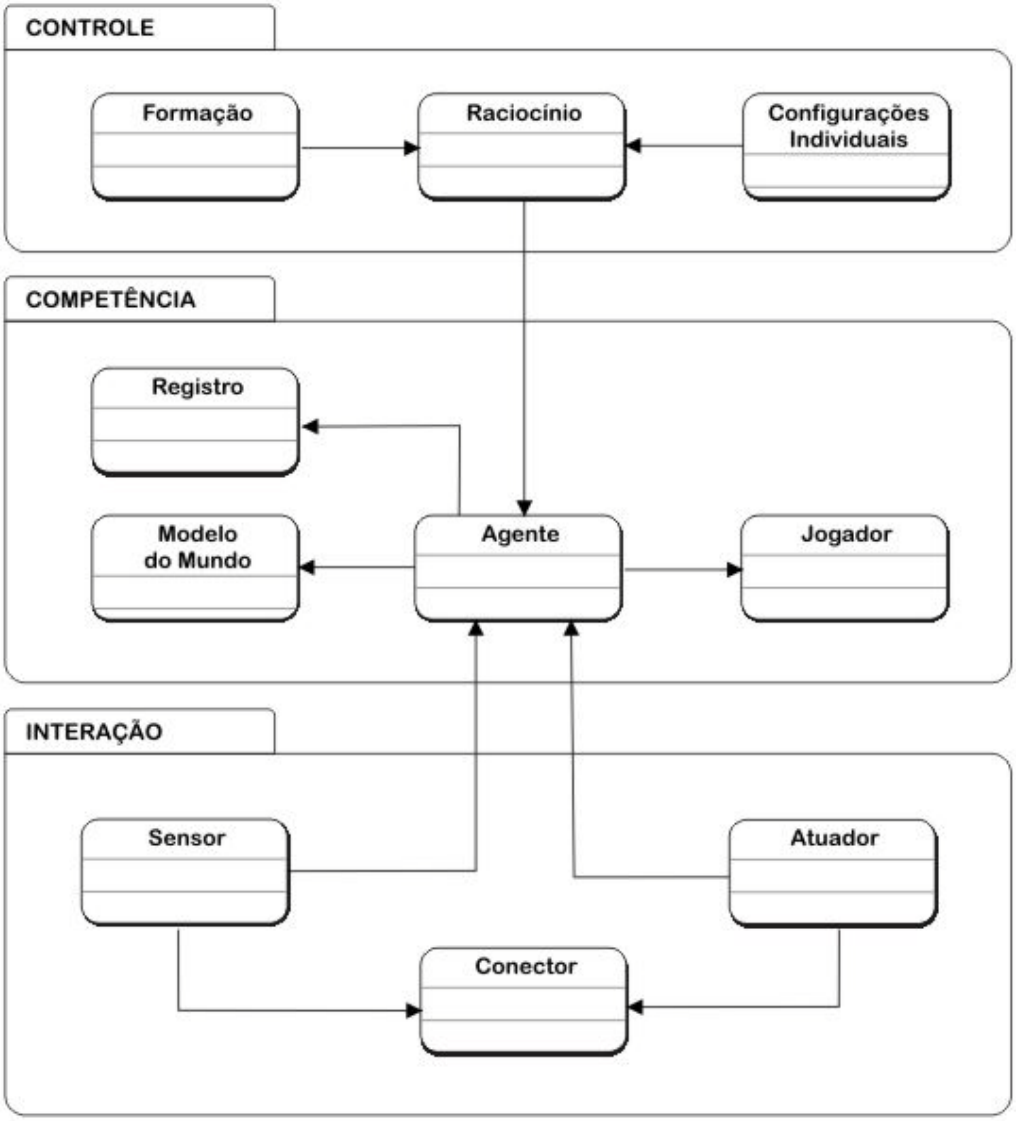


**Figura 08:** Pacote e componentes da arquitetura do time

A seguir apresentamos o papel de cada um dos componentes quem compõe o agente:

- **Conector:** componente responsável por criar uma conexão socket UDP com o servidor, contendo métodos para enviar e receber mensagens por meio dessa conexão;
- **Sensor:** esse componente processa as mensagens recebidas do servidor, particiona e envia as informações extraídas para o componente Agente que trata de atualizar o Modelo do Mundo;
- **Atuador**: responsável por executar os comandos que o jogador deve realizar, assim que foi recebido do componente Agente.

- **Agente:** Gerencia os a interação entre as camadas e contem métodos para controle do processamento paralelos dos componentes Sensor, Atuador e Raciocínio;
- **Modelo do Mundo**: responsável por criar a representação atual do jogo, do ponto de vista observado pelo agente, determinando-se a posição do agente, da bola e de suas respectivas velocidades. E componente contém informações sobre todos os objetos na simulação, como demarcações do campo, jogadores, bola e posição do gol;
- **Jogador:** esse componente contém todas as informações necessárias para um perfil de jogador, como interceptar a bola ou chutar a bola para uma determinada posição do campo;
- **Raciocínio**: contém métodos para resolver sobre a possibilidade da melhor ação em uma determinada situação. É o cérebro do Agente;
- **Formações:** esse componente contém informações sobre as possíveis formações do time e do método para determinar a posição estratégica;
- **Configurações individuais**: esse componente contém os valores de muitos parâmetros do jogador, tal como a influência para resolver algum processo do agente e de definir métodos para receber e atualizar esses valores;
- **Registro**: Registra as ações durante a execução do Agente. Possui controle de verbosidade para saber o que deve armazenar.

A arquitetura de três camadas, sendo uma camada hierárquica de competências, se apresenta como uma forma capaz de realizar o processamento paralelo dos dados, um esquema flexível de sincronização do ambiente, com métodos para localização de objetos e estratégias.

Diferentemente do modelo apresentado por RIBEIRO (2010) os componentes foram organizados em pacotes que representam as camadas do agente (controle, competência, interação). Houveram também a inclusão de componentes Formação e Configurações Individuais, que auxiliam o componente Raciocínio.

O processamento do componente raciocínio se faz de forma paralela, para que haja o processamento das predições ao mesmo tempo em que as ações são executadas, como veremos mais à frente no tópico 4.4.

Além destes componentes principais, foram desenvolvidas várias outras classes auxiliares que são utilizadas pelos diversos componentes. As classes e suas funcionalidades são:

- **Parser:** Esta classe contem vários métodos estáticos para transformação de mensagens recebidas do Servidor RoboCup pelo através do componente Sensor. Os métodos podem isolar partes das mensagens, processar e converter para tipos básicos ou objetos de classes internas do sistema.
- **ConfiguracaoInicial:** Esta classe contem vários parâmetros utilizados pelo Servidor RoboCup, como versões do protocolo de comunicação, limites entre intervalos de parâmetros de comandos (ex: Incremento máximo de stamina – vigor – ou velocidade máxima de um jogador , entre outros).
- **TipoJogador:** Esta classe contem configurações básicas de diferentes tipos de Jogador que se podem ter em campo. Ela cria uma abstração para os tipos de jogadores de futebol com posicionamentos e comportamento.
- **Local:** Esta classe contém as coordenadas x e y denotando as posicoes em campo. Contem métodos que tratam de posições e cálculos entre posições. Trata também dos posicionamentos em campo por regiões e quadrantes, que serão apresentadas mais à frente neste capítulo.
- **Linha:** Representação geométrica de uma linha em campo. Contem métodos que tratam dos cálculos geométricos para esta forma.
- **Retângulo:** Representação geométrica de um retângulo em campo. Contem métodos que tratam dos cálculos geométricos para esta forma.
- **Círculo:** Representação geométrica de um círculo em campo. Contem métodos que tratam dos cálculos geométricos para esta forma.
- **Geometria:** Implementa vários métodos que tratam dos cálculos geométricos usados.

## *4.4. Multiprocessamento do SMA*

Neste sistema multiagente cada agente inteligente representa um processo rodando na maquina. Em uma competição oficial de um Campeonato de Futebol Simulado 2D cada time pode dispor de uma certa quantidade de máquinas para executar seus jogadores. A Figura 09 ilustra este modelo.

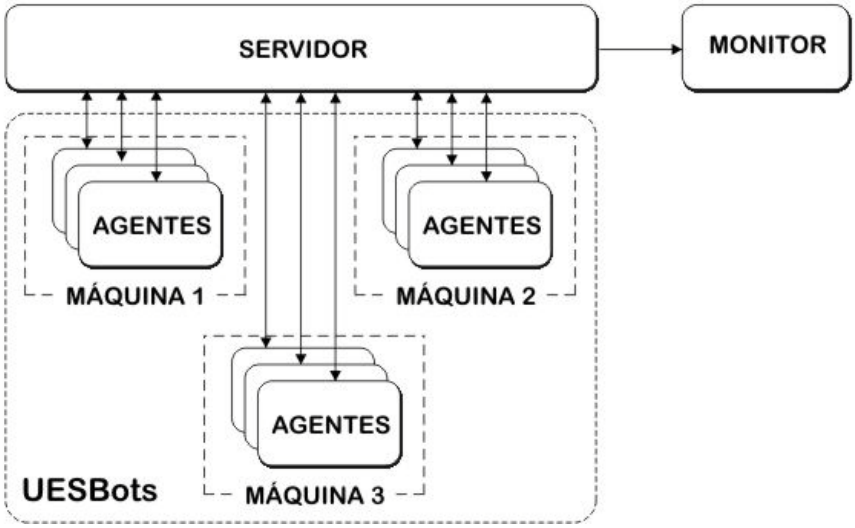


**Figura 09:** Modelo do multiprocessamento do SMA UESBots

Devemos lembrar também que uma das regras do RoboCup é o isolamento de cada agente, não podendo haver comunicação lateral e a única forma de comunicação é através do Servidor RoboCup com os comandos de falar e ouvir.

Para facilitar a inicialização dos agentes foi criada a classe Time que representa a execução de um numero N (configurável) de agentes na máquina. Esta classe trata do multiprocessamento dos agentes e inicializa os processos na máquina.

Como configuração inicial esta classe lê um arquivo de configuração que contem os seguintes parâmetros, exibidos abaixo na Tabela 09.

**Tabela 09:** Parâmetros iniciais para execução do SMA UESBots

| | |
|---|---|
| ***Host*** | Endereço da máquina onde está sendo executada o Servidor RoboCup |
| ***Port*** | Porta de acesso utilizado pelo socket para conexao UDP pelo componente Conector |
| ***Nome*** | Nome do time que está sendo inicializado (ex: UESBots) |
| ***Jogadores*** | Número de agentes jogadores que devem ser inicializados |
| ***Goleiro*** | Configura se o agente jogador do tipo goleiro deve ser inicializado |
| ***Esquema*** | Nome do esquema tático utilizado pelo time. Necessário para inicializar as configurações sobre posicionamento e áreas de atuação dos jogadores em campo. Veremos mais detalhadamente informações sobre os esquemas táticos mais à frente neste capítulo. |

## *4.5. Threading*

A classe Agente recebe as configurações iniciais da classe Time. Quando o agente é iniciado são criados dois threads[3] (processamentos concorrentes em uma mesma aplicação) para o Sensor e Raciocínio permitindo assim que a execução corrente do Agente possa continuar enquanto há o processamento paralelo das percepções no Sensor e a escolha da melhor ação atual e de ações futuras (predição) pelo Raciocínio. A Figura 10 mostram as relações entre as classes e suas interfaces, bem como métodos da interface IAgente implementados no Componente Agente.

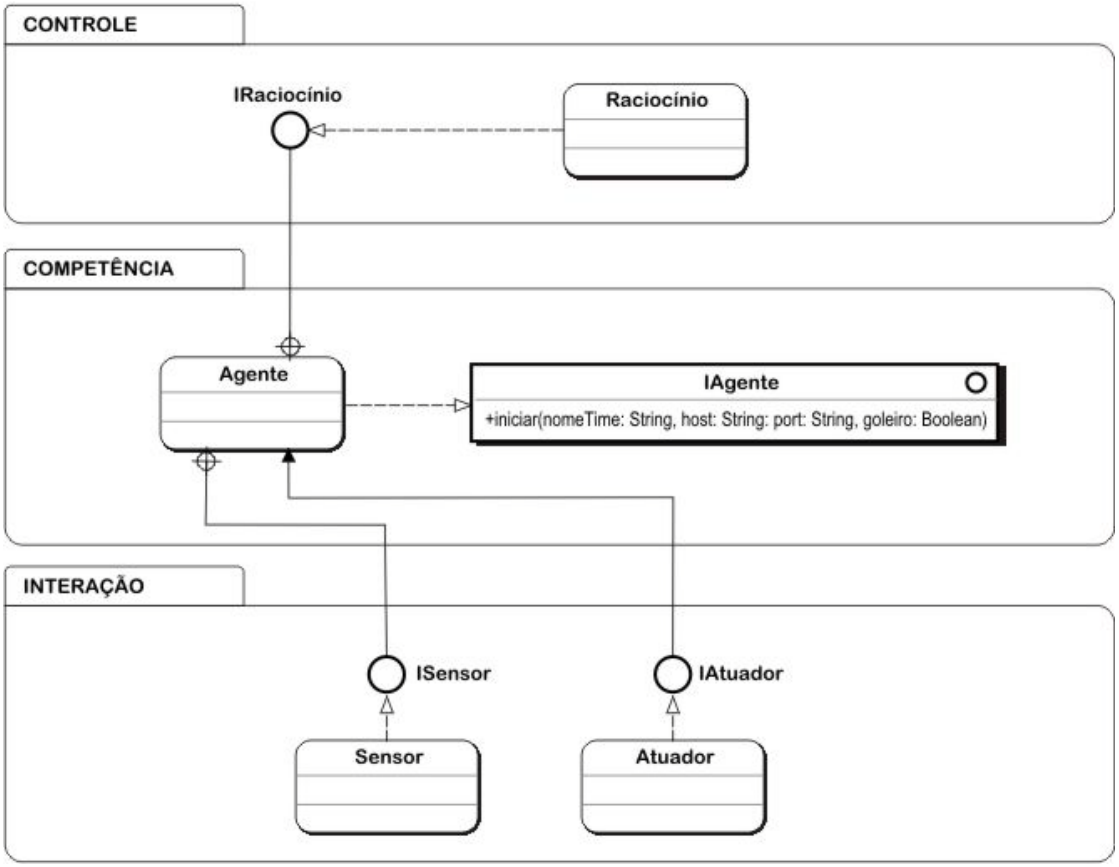


**Figura 10:** Modelo do componente Agente: pacotes, interfaces e relações

O fluxo de atividade dos Threads exibido na Figura 11 é dividido em três processos concorrentes. No primeiro o Sensor mantém a execução contínua de funções que recebem informações do simulador. Já o segundo corresponde à execução dos demais componentes referentes e execução de ações. Um terceiro corresponde ao processamento do raciocínio, predição de ações e cognição.

[3] Linha de execução (thread em inglês) é uma forma de um processo dividir a si mesmo em duas ou mais tarefas que podem ser executadas concorrentemente.

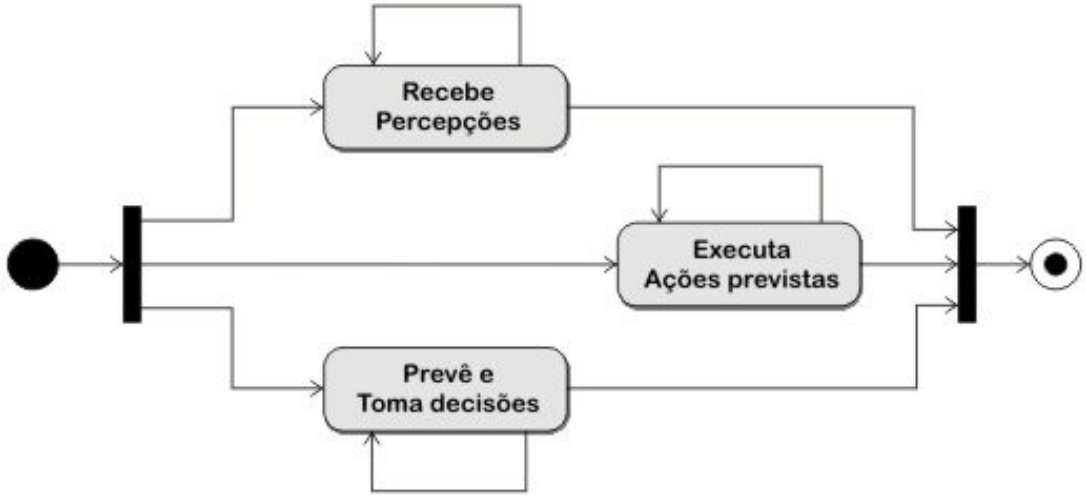


**Figura 11:** Fluxo de threads do sistema

## *4.1. Enumerações*

Algumas enumerações auxiliares modeladas por RIBEIRO (2010) foram mantidas por se julgar suficientes para a execução do desenvolvimento dos agentes. A Tabela 10 mostra estas enumerações.

**Tabela 10:** Enumerações auxiliares

| | |
|---|---|
| ***Lado*** | Esquerdo, Direito |
| ***Distâncias*** | Margem Chute, Perto, Muito Perto, Intermediária, Longe, Muito Longe, Indeterminado |
| ***VisaoAngulo*** | Estreito, Normal, Amplo |
| ***Direcoes*** | N, S, L, E, NO, NE, SO, SE |
| ***VisaoQualidade*** | Alta, Baixa |
| ***Velocidade*** | Parado, Muito Lento, Lento, Normal, Rápido, Muito Rápido |
| ***TipoObjeto*** | Bola, Bandeira, Jogador, Linha |
| ***Força*** | Muito Fraco, Fraco, Normal, Forte, Muito Forte |
| ***RegiaoH*** | Meio, Inter, Gol |
| ***RegiaoV*** | Lateral Esquerda, Lateral Direita, Central Esquerda, Central Direita |
| ***QuadranteX*** | E0, E1, E2, E3, D0, D1, D2, D3 |
| ***QuadranteY*** | A, B, C, D, E, F |

## *4.1. O Componente Jogador*

Embora o Sensor e Atuador estejam conectados ao servidor, eles são apenas os meios de contato do agente com o ambiente e não tem autonomia. Portanto é necessário que exista um modulo principal capaz de unir os demais. Na Figura 12 observamos detalhes do componente Jogador. Suas funcionalidades são:

- Atualizar o estado do jogo mantido pelo ModeloMundo;
- Recorrer ao componente Raciocínio para tomada de decisões;
- Enviar as ações tomadas para do componente Atuador;
- Enviar requisições para o componente Sensor;
- Manter informações do jogo como o placar, nome do seu time, tempo de jogo em ciclos, mensagem do árbitro para os estados do jogo, etc.

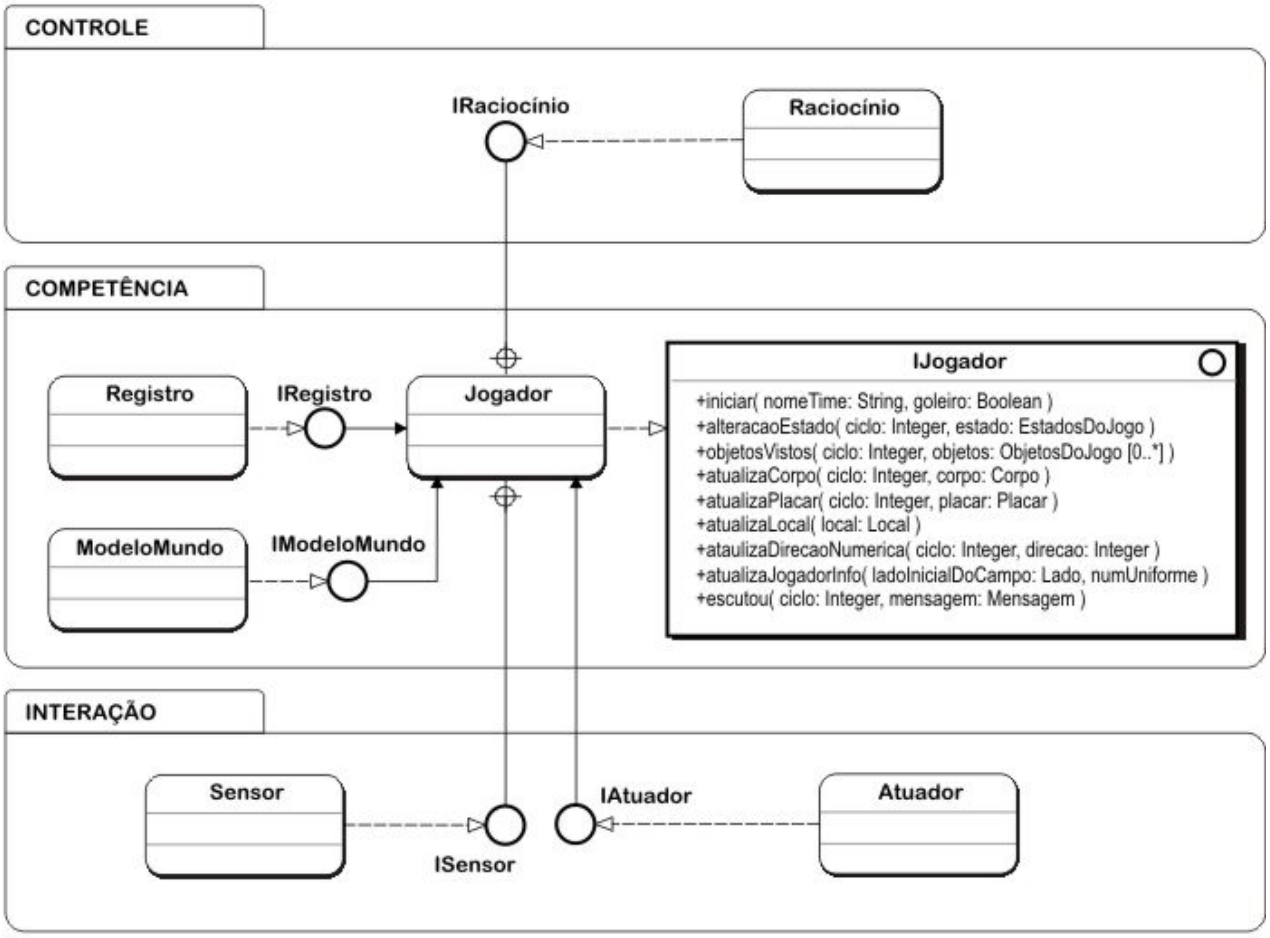


**Figura 12:** Componente Jogador

A Tabela 11 explica os métodos da interface IJogador implementados na classe Jogador.

**Tabela 11:** Métodos da Classe Jogador

| | |
|---|---|
| **Iniciar** | Executa funções iniciais do componente, em seguida armazena o nome do time e se o jogador tem o papel de goleiro |
| **alteracaoEstado** | A alteração de estado é utilizada para informar ao jogador, o estado do jogo determinado pelo juiz da partida, como: iniciada, não iniciada, finalizada, cobrança de escanteio, etc |
| **objetosVistos** | Recebe grupos de objetos vistos pelo agente, ou objetos específicos que foram requisitados ao Sensor |
| **atualizaCorpo** | Recebe os dados do corporais do agente |
| **atualizaPlacar** | Atualiza o placar do jogo, quando um gol é feito |
| **atualizaLocal** | Atualiza o local atual do agente |
| **atualizaDireçãoNumerica** | Atualiza a direção do corpo do jogador, que é necessária para execução de quase todos os comandos do Atuador |
| **atualizaJogadorInfo** | Atualiza informação do jogador como o lado inicial que o time entrou em campo e o número que o jogador recebeu |
| **Escutou** | Recebe dados de comunicação dos agentes contendo uma mensagem de tamanho máximo de 512 caracteres, o seu remetente e sua direção em relação ao ouvinte caso seja um jogador |

## *4.2. Modelo de Mundo*

Para o agente se comportar inteligentemente é importante manter um modelo de mundo que descreva o corrente estado do ambiente. O agente pode usar este modelo para decidir a melhor ação possível na situação. Este modelo pode ser visto como uma memória do estado do jogo baseada em percepções anteriores. Ele possui vários tipos de informação a respeito de todos os objetos no campo, assim como diversos métodos que usam esta informação para derivar previsões de próximas ações.

Assim sendo uma ferramenta essencial para agentes cognitivos decidirem que ações serão mais apropriadas em determinado estado do ambiente. No conjunto de

componentes do agente o mundo é representado pelo ModeloMundo. Esse permite que cada agente jogador possa construir internamente o seu próprio modelo do mundo se baseando em percepções visuais, auditivas e corporais.

A Figura 13 mostra a hierarquia de classes apta a conservar os objetos do jogo referentes a elementos do ambiente. Os objetos do jogo são classes que através de generalização, herdam os atributos de suas antecessoras na hierarquia.

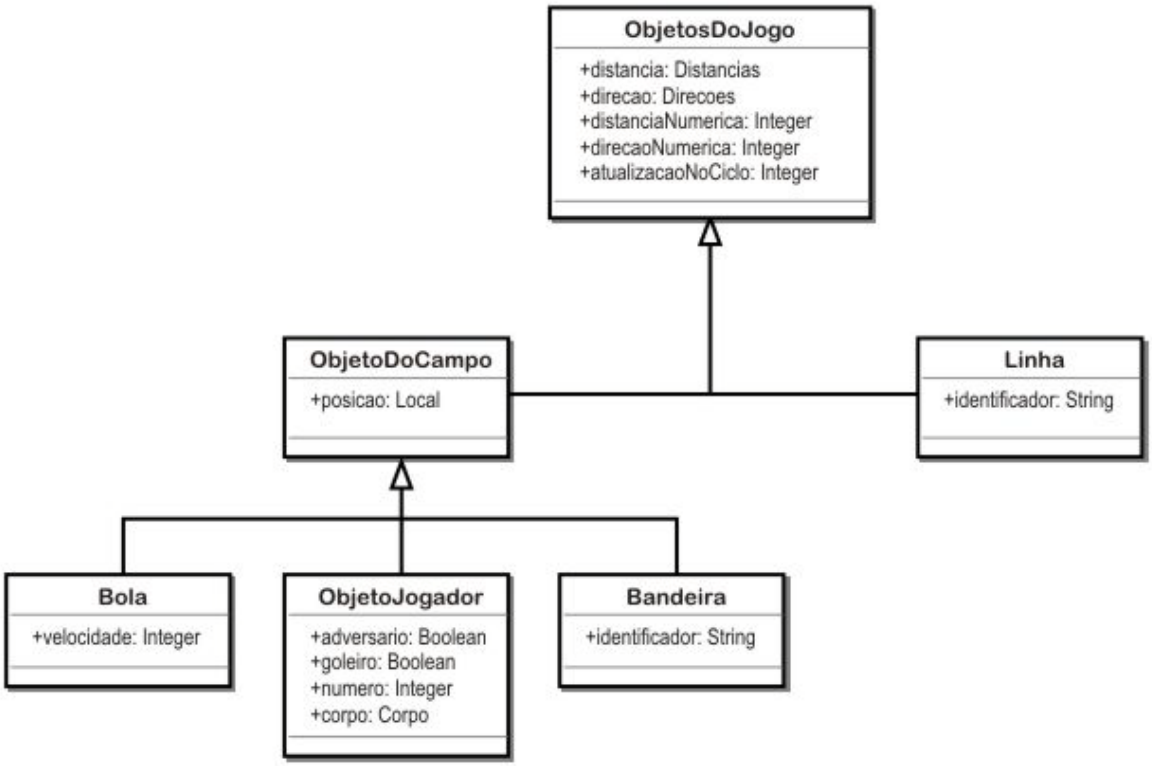


**Figura 13:** Hierarquia de classes dos objetos do jogo

Existem dois componentes que mantém relações com o ModeloMundo. O Jogador, que é responsável por adicionar informações do mundo selecionadas entre as percepções do agente e o componente Raciocínio que utiliza essas informações.

### *4.2.1. O Campo de Futebol Virtual*

No desenvolvimento deste trabalho foi mantida a divisão proposta por RIBEIRO (2010) que particiona o campo em regiões e quadrantes como visto da Figura 14.

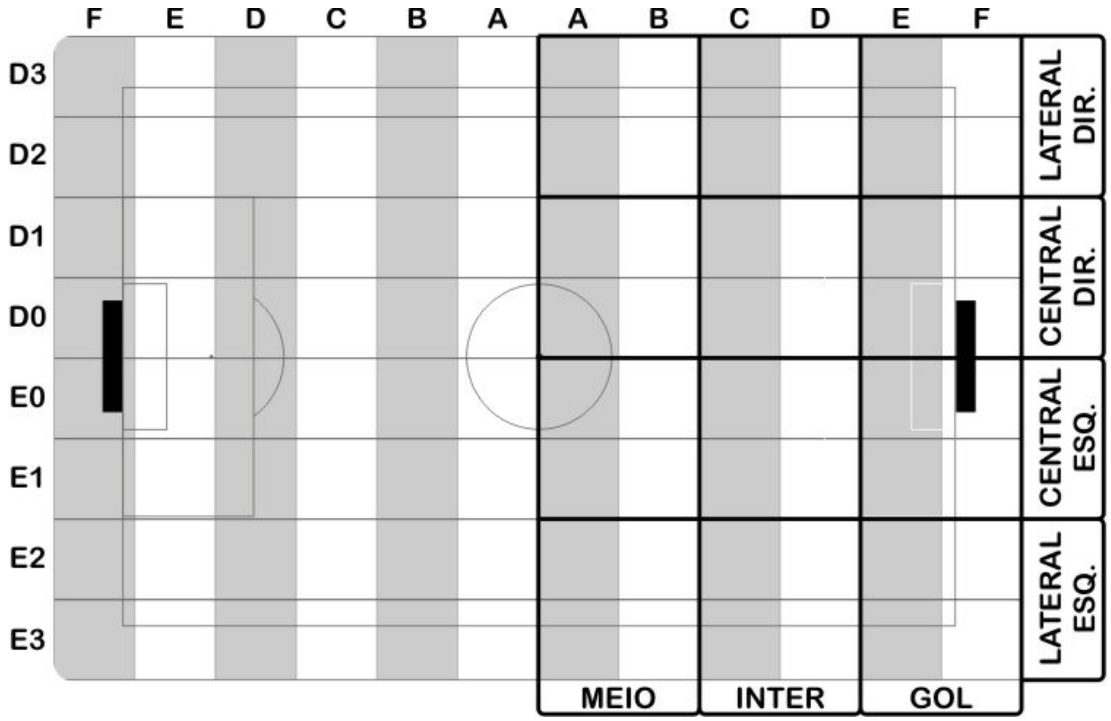


**Figura 14:** Hierarquia de classes dos objetos do jogo

O campo é dividido em regiões e quadrantes onde as regiões servem para delimitar áreas onde o jogador deverá atuar, ou seja, nessas áreas o jogador deverá fiscalizar e marcar os adversários e sempre buscar a localização da bola, a fim de receber passes dos companheiros. Já os quadrantes são utilizados para movimentação do jogador, e tem o objetivo de facilitar as decisões do sistema de inteligência, de forma que não será necessário trabalhar com valores numéricos, pois tornariam o raciocínio pouco eficiente.

Na Figura 14 os valores F, E, D, C, B e A são correspondentes a abscissa e os valores de D3-D0 e E0-E3 a ordenada do campo virtual. A combinação deles representa quadrantes para cada lado do campo.

As regiões tem uma área equivalente a de quatro quadrantes, definidas com as combinações de: meio, intermediário e gol na horizontal e na vertical por: central esquerda/direita e lateral esquerda/direita.

Quando um comando de movimento é requisitado. O Atuador devera posicionar o agente no local mais adequado nos limites do quadrante especificado, então algum comando como “interceptarBola” pode ser utilizado.

## 4.3. Sensor, Atuador e Conector

As rotinas de baixo nível para comunicação entre o servidor e o SMA UESBots utilizadas pelos componentes Sensor, Atuador através do componente Conector foram mantidas do trabalho base realizado por RIBEIRO (2010).

O Sensor representa a interface de entrada, o componente Sensor é responsável por traduzir os dados das percepções recebidas para informações de alto nível compostas por objetos que o agente seja capaz de lidar. Através dessas informações o agente pode saber das condições de seu corpo e utilizando suas percepções visuais ser capaz de construir e atualizar um modelo do estado do jogo. Esses conhecimentos são essenciais para que o agente possa criar planos ou tomar decisões.

O atuador representa a interface de saída e é o mecanismo que traduz as ações de alto nível definidas pelo agente para comandos de baixo nível que são enviados para o servidor sob forma de texto, seguindo as normas do protocolo do Servidor RoboCup discutidas no Capítulo 2, Tópico 2.5.3.

## 4.4. Raciocínio

O componente Raciocínio é responsável pela inteligência do jogador. Ele deve ser capaz de avaliar a cada momento as condições dos agentes e do ambiente do jogo, lida com as prioridades de comportamentos para os diferentes tipos de jogadores (goleiro, zagueiro, atacante, etc.), e ainda ser apto a criar, aplicar e modificar estratégias para satisfazer as metas da equipe.

### 4.4.1. Predição de Ações

Métodos foram implementados para predição de futuros estados do mundo baseados em percepções anteriores. Estes métodos são importantes para o processo de seleção de ação. Existem métodos de predição de baixo nível, implementados expandindo as ações do componente ModeloMundo, tais como prever informações a respeito de futuras posições de objetos em campo. Já os métodos de predição de alto nível como prever o sucesso de determinada estratégia ou o comportamento de um jogador adversário não foram implementados neste trabalho.

Os métodos implementados são divididos em três categorias:

- **Previsão do estado do agente após executar determinada ação**: Os métodos recebem um determinado comando de ação e retorna a previsão das informações do agente. A Tabela 12 mostra alguns destes métodos que fazem previsão de valores.

**Tabela 12:** Métodos de predição de estados do jogador

| | |
|---|---|
| **preverPosicao** | Prevê a posição do jogador baseado nas posições e velocidade nos últimos ciclos |
| **preverVelocidade** | Prevê a velocidade que o jogador vai ter baseado em posições de ciclos anteriores |
| **preverAnguloCorpo** | Prevê o ângulo do corpo do jogador baseado nos últimos ângulos do corpo em ciclos anteriores |
| **preverAnguloCabeca** | Prevê o ângulo da cabeça do jogador baseado nos últimos ângulos da cabeça em ciclos anteriores |
| **preverVigor** | Baseado nas ultimas ações do jogador em ciclos anteriores é possível prever o vigor que ele terá |

- **Previsão do estado outros objetos dinâmicos no campo:** É muito difícil para jogadores visto que o agente não tem conhecimento de quais ações estes pretendem tomar. Em geral previsões a respeito de outros jogadores não são feitas e os métodos implementados (*preverVelocidadeBola* e *preverPosicaoBola*) têm foco no objeto Bola, que é mais fácil de prever visto que o movimento da bola é menos aleatório.
- **Previsão de tempo que um jogador levará para chegar a determinada posição no campo:** Este método implementado (*preverCicloChegada*) recebe a posição alvo e o jogador como argumentos e retorna o número estimado de ciclos que o jogador vai precisar para chegar até a posição desejada. Este método faz a previsão baseada na distância máxima que o jogador pode andar em um ciclo (ignorando as variáveis de ambiente Ruido e Vento e também o tempo que o jogador leva para virar para o

ângulo correto). Note que o resultado é somente uma estimativa aproximada.

### *4.4.2. Habilidades Individuais dos Jogadores*

As habilidades dos jogadores implementadas no time UESBots incluem: ir até determinado ponto, chutar uma bola para uma direção desejada, driblar, interceptar uma bola, marcar adversários, etc. Estas habilidades foram divididas em três níveis que juntas formam uma hierarquia de habilidades, como pode ser visto na Figura 15.

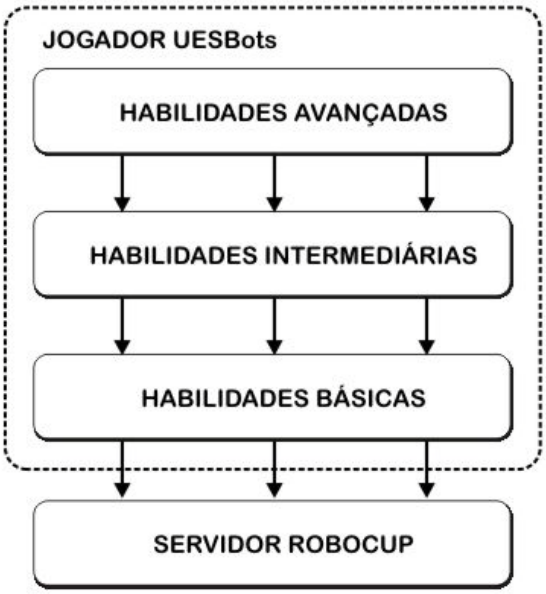


**Figura 15:** Hierarquia habilidades de um jogador

As camadas são hierárquicas no sentido em que cada habilidade de uma camada utiliza habilidades da camada inferior para gerar o comportamento desejado.

As habilidades básicas podem ser consideradas como os comandos básicos recebidos pelo Servidor RoboCup. O nível intermediário possui habilidades baseadas nas habilidades básicas e não tem que lidar mais com o formato de comunicação de comandos com o servidor. Na ultima camada se encontram as habilidades avançadas baseadas nas intermediárias.

#### *4.4.2.1. Habilidades básicas*

Como existe um mapeamento direto entre as habilidades básicas e os comandos básicos, estas habilidades não contêm decisões complexas. A Tabela 13 lista os métodos implementados.

**Tabela 13:** Habilidades básicas do jogador

| | |
|---|---|
| **alinharPescocoComCorpo** | Alinha o pescoço do jogador com o corpo enviando o comando básico *turn_neck* com ângulo zero. *(turn_neck 0)* |
| **virarCorpoParaPosicao** | Vira o corpo do jogador para uma posição do campo. A posição pode ser um Quadrante, Regiao ou coordenadas x, y |
| **virarCostasParaPosicao** | A única diferença para o método anterior é que a referência usada são as costas do jogador. Auxilia principalmente o trabalho do goleiro que pode se mover de volta para a área do gol sem perder o ângulo de visão do campo. |
| **virarPescocoParaPosicao** | Vira o pescoço do jogador para uma posição no campo. A posição pode ser um Quadrante, Regiao ou coordenadas x, y |
| **procurarBola** | Permite que o jogador procura pela bola quando não se tem visão direta para ela |
| **arrancarParaPosicao** | Faz com que o jogador realize um movimento com determinada velocidade. É preciso entender que o jogador só pode basicamente se movimentar para frente e para trás. |
| **pararBola** | Faz com que o jogador pare a bola numa determinada posição. Utiliza o comando chutar, calculando a força necessária para que o chute pare a bola. |
| **dominarBola** | Mantem a bola próxima ao corpo do jogador. Utiliza o comando chutar, calculando a força necessária para manter a bola perto de si. |
| **acelerarBolaVelocidade** | Permite que o jogador chute a bola de tal forma que esta ganhe uma velocidade especifica ao final |
| **pegarBola** | Permite que o jogador pegue a bola. Usado pelo goleiro. |
| **Falar** | Permite ao jogador falar uma mensagem aos jogadores ao seu redor |

#### *4.4.2.2. Habilidades Intermediárias*

Estas habilidades utilizam os métodos de habilidades básicas para realizar uma ação com comportamento desejado. Em forma simplista ela utiliza as informações

correntes do ambiente (ModeloMundo) para encontrar os melhores valores a serem passados para os métodos básicos. A Tabela 14 descreve estas habilidades.

**Tabela 14:** Habilidades intermediárias do jogador

| | |
|---|---|
| **virarCorpoParaObjeto** | Vira o corpo do jogador para a posição do campo em que se encontra o objeto alvo. Utiliza o método básico **virarCorpoParaPosicao** |
| **virarPescocoParaObjeto** | Vira o pescoço do jogador para a posição do campo em que se encontra o objeto alvo. Utiliza o método básico **virarPescocoParaPosicao** |
| **moverParaPosicao** | Permite que o jogador se movimente para qualquer posição do campo. Uma vez que o jogador só se move basicamente para frente ou para trás, este método utiliza os métodos **arrancarParaPosicao, virarCorpoParaPosicao**, **virarCostasParaPosicao** |
| **aproximarDaBola** | Permite que o jogador se movimente até o mais próximo possível da bola |
| **chutarBolaParaPosicao** | Permite que o jogador chute uma bola até determinada posição do campo a uma determinada velocidade |
| **virarComBola** | Permite que o jogador vire o corpo mantendo a bola próxima ao corpo. Utiliza **pararBola, dominarBola, virarCorpoParaPosicao** |
| **moverParaPosicaoNaReta** | Permite que o jogador se movimente até determinada posição de uma reta qualquer. Permite a fabricação de jogadas. |

#### *4.4.2.3. Habilidades Avançadas*

Estas habilidades utilizam os métodos de habilidades intermediárias para realizar uma ação com comportamento desejado utilizando como entrada as informações correntes do ModeloMundo. A Tabela 15 descreve estas habilidades.

**Tabela 15:** Habilidades avançadas do jogador

| | |
|---|---|
| **interceptarBola** | Permite que o jogador intercepte a bola em qualquer distancia. |
| **driblar** | Permite ao jogador se manter com a bola enquanto realiza movimentos. O jogador chuta a bola em determinada velocidade numa direção |

| | |
|---|---|
| | especifica e se aproxima dela outra vez |
| **passarBola** | Permite que o jogador passe a bola para um outro jogador do mesmo time |
| **passarBolaAFrente** | Permite ao jogador passar a bola para uma posição à frente de outro jogador para que ele receba. Difere do **passarBola** pois o jogador não a recebe diretamente. |
| **lancarEmProfundidade** | Permite ao jogador dar um tipo de passe mais avançado lançando a bola entre adversários para uma posição em que outro jogador possa recebê-la |
| **derrotarAdversario** | Permite que o jogador possa lançar a bola numa posição ótima à frente do adversário de forma que ele possa passar pelo adversário sem a bola e novamente buscá-la do outro lado. |
| **limparBola** | Permite que o jogador possa chutar a bola para determinada área do campo. É útil quando o jogador não pode driblar ou passar a bola para um companheiro e não resta outra alternativa |
| **marcarOponente** | Permite que o jogador marque um determinado oponente, guardando ele um-a-um com o propósito de minimizar a utilidade do adversário para o time adversário |
| **defenderLinhaDeGol** | Permite ao jogador (usualmente goleiro) defender seu próprio gol. |

No desenvolvimento deste trabalho nem todas as habilidades avançadas foram implementadas, tais como **passarBolaAFrente**, **lancarEmProfundidade**, **derrotarAdversario** e **marcarOponente**.

### *4.4.3. Estratégia de Time*

Aplicar com eficácia as informações provenientes das percepções do agente na tentativa de explorar ou criar condições favoráveis faz parte do planejamento estratégico do time.

Existem duas dinâmicas envolvidas no futebol de robôs: jogo de equipes (objetivo global) e o jogo dos jogadores de uma mesma equipe (objetivos locais).

#### *4.4.3.1. Objetivos Globais*

O objetivo do jogo das equipes é fazer com que uma determinada equipe consiga fazer mais gols do que levar. Tendo como resultados possíveis: Ganhar, Empatar ou Perder.

Basicamente foram definidos dois estados que as equipes podem assumir:

- **COM BOLA:** onde o objetivo é fazer gol;
- **SEM BOLA:** o objetivo é não tomar gol.

As escolhas que a equipe pode fazer em um dado momento do jogo são: **ATACAR** (tentar se posicionar no campo do adversário) ou **DEFENDER** (tentar se manter em seu próprio campo).

Os parâmetros utilizados para se determinar o estado atual de uma equipe são:

- Distância dos jogadores em relação à bola;
- Velocidade dos jogadores;
- Velocidade da bola.

Portanto os possíveis estados que o time pode assumir: COM BOLA (quando um robô da equipe controlada está mais próximo da bola do que qualquer outro robô da equipe adversária) e SEM BOLA (quanto um robô da equipe adversária está mais próximo da bola do que qualquer robô da equipe controlada). Em ambos os estados, o jogador pode ATACAR ou DEFENDER.

O pseudocódigo do objetivo global pode ser visto no Quadro 01.

```
#determina a estratégia global para equipe
inicial = cicloAtual; #ciclo atual do jogo
final = tempoResposta #periodo em ciclos

if dadosTime.estado = COM_BOLA:
        dadosTime.estrategia = ATAQUE;

elif dadosTime.estado = SEM_BOLA and inicial < final:
        dadosTime.estrategia = DEFESA;

elif dadosTime.estado = SEM_BOLA and inicial >= final:
        dadosTime.estrategia = ATAQUE;
```

**Quadro 01:** Algoritmo para determinar a estratégia global

Basicamente, essas estratégias, ATACAR ou DEFENDER, definem um esquema tático que os robôs deverão assumir em grupo. Mais à frente, no tópico 4.4.3.3, veremos outras adições ao esquema tático como formações e áreas de atuação em campo.

#### *4.4.3.2. Objetivos Locais*

Para ampliar a estratégia colaborativa do time foram definidas algumas classes para jogadores. Cada classe tem um papel definido no jogo e é responsável por uma atitude comportamental diferente de acordo com a área de atuação. Estas classes são:

- **Goleiro:** responsável por defender e impedir que a bola ultrapasse a linha de gol. Seu comportamento é defensivo;
- **Zagueiro:** sua função básica é defensiva, este jogador também deve ter um bom senso de cobertura de campo;
- **Lateral (Direito e Esquerdo):** são os responsáveis pela construção das jogadas e tem a tarefa de marcar e atacar, portando um comportamento misto;
- **Meio de Campo:** Tem a tarefa de abrir espaços na área adversária para a penetração de seus companheiros e quando acionado exerce as ações de finalização;
- **Atacante:** Exerce as ações de finalização

#### *4.4.3.3. Formações e Posicionamento Estratégico*

Foi implementado um esquema de áreas de atuação para cada tipo de classe de jogador definida. O arquivo de configuração inicial do SMA UESBots configura o nome do esquema tático que será usado. Através deste parâmetro o esquema tático específico é carregado pelo sistema onde há definições de posicionamento e de regiões do campo que cada tipo de jogador pode assumir defensivamente ou ofensivamente. O Quadro 02 mostra o formato do arquivo de configuração de esquema tático.

```
[formacao]
nome=442
ordem=G,Z1,Z2,LD,LE,M1,M2,M3,M4,A1,A2

[G] tipo=Goleiro iniX=-50 iniY=0
[Z1] tipo=Zagueiro iniX=-35 iniY=-10
[Z2] tipo=Zagueiro iniX=-35 iniY=10
[LD] tipo=LateralDireita iniX=-30 iniY=-25
[LE] tipo=LateralEsquerda iniX=-30 iniY=25
[M1] tipo=MeioCampo iniX=-21 iniY=0
[M2] tipo=MeioCampo iniX=-12 iniY=-15
[M3] tipo=MeioCampo iniX=-12 iniY=15
[M4] tipo=MeioCampo iniX=10 iniY=0
[A1] tipo=Atacante iniX=-7 iniY=-7
[A2] tipo=Atacante iniX=-7 iniY=7
```

**Quadro 02:** Formato do arquivo de configuração de esquema tático

Neste arquivo pode-se ver o nome do esquema tático empregado e a ordem de entrada dos jogadores que corresponde ao numero das camisas distribuídas pelo Servidor RoboCup, sendo assim o Goleiro representado pela letra G recebe a camisa número 1. Em seguida pode-se ver a configuração de posição inicial de cada jogador. Esta posição é assumida antes do início da partida e após cada gol.

Complementarmente a esta configuração inicial foram definidos arquivos (formato texto) complementares para cada tipo de classe onde se definem as áreas de atuação de cada jogador. O texto é formatado como uma matriz, onde cada região é representada por uma interseção linha x coluna marca-se a região do campo onde o jogador tem prioridade de defesa (letra D) e ataque (letra A). A letra X representa uma região fora da área de atuação. Um exemplo é mostrado no Quadro 03.

```
XXXXXX
XXDAAA
XXDAAA
XXXXXX
```

**Quadro 03:** Formato do arquivo das regiões de atuação do jogador tipo Meio Campo

Este arquivo mostrado no Quadro 03 representando as áreas de atuação pode ser mais bem entendido através da Figura 16.


**Figura 16:** Representação gráfica do arquivo de configuração de regiões de atuação

A representação gráfica do esquema tático do jogo após o carregamento de todos os arquivos com as configurações de posicionamento é mostrada na Figura 17.

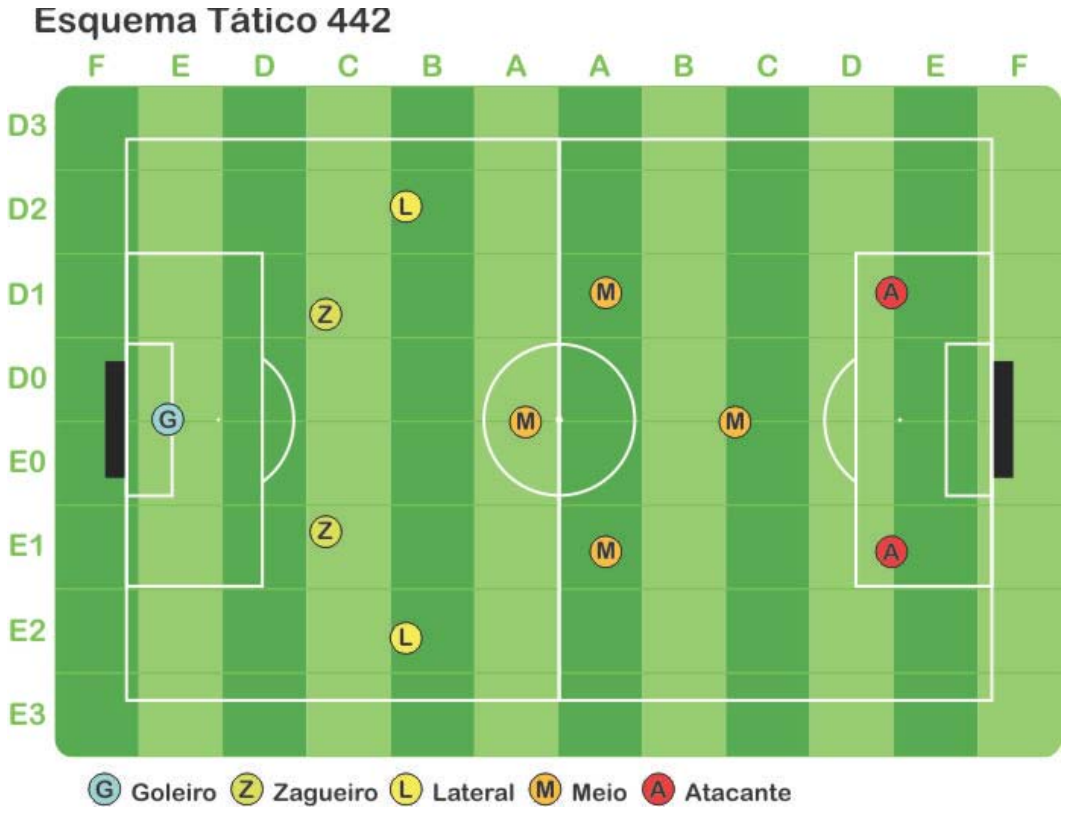


**Figura 17:** Representação gráfica do esquema tático carregado

#### *4.4.3.4. Audição e Fala*

Como os agentes não podem se comunicar diretamente a utilização dos comandos de fala e audição são essenciais para a ampliar a visão de mundo dos jogadores.

Durante o jogo cada jogador no time determina quando é o melhor momento de comunicar as informações contidas no ModeloMundo. Esta decisão depende da posição da bola no campo e também em qual papel o jogador atua dentro da formação tática do time. Desde que o principal objetivo da comunicação é aumentar a quantidade de informação a respeito do ambiente é importante comunicar uma boa visão do campo e dos oponentes para o máximo de jogadores. Em especial a parte do campo onde se encontra a bola deve estar bem atualizada no ModeloMundo do jogador.

Outros exemplos de mensagens enviadas pelos métodos de fala são as requisições ou informações relevantes para a execução de uma ação pelo agente, como:

- ***(say "tolivre"):*** Indicando que o agente que emitiu a mensagem está livre para receber um passe, ou;
- ***(say "aparece"):*** Quando um jogador precisa do auxilio de algum companheiro de time.

## *4.5.Avaliação*

Foram realizados testes onde o time UESBots enfrentava outro time de igual implementação (outra instância do SMA UESBots) e também contra o UvA Trilearn.

Em partidas contra o próprio time UESBots ficou claro que dividir o campo em regiões e quadrantes definindo as áreas de atuação de cada jogador por regiões, geram muitas áreas de sobreposição quanto à atuação dos jogadores. No exemplo de área de atuação do jogador de tipo MeioCampo, mostrado no capítulo anterior, vários jogadores do mesmo tipo avançavam em direção à bola atrapalhando a estratégia de time. Mesmo depois de implementado o algoritmo matemático de distância euclidiana para maximizar a disposição dos jogadores em relação à bola o resultado não foi o esperado.

Uma sugestão seria aumentar o particionamento do campo. Um exemplo que parece ser acertado é o modelo proposto por KHASHABI (2009) que utiliza um

particionamento do campo por triangulação, conhecido como Triagunlação Delaunay, conforme mostrado na Figura 18.

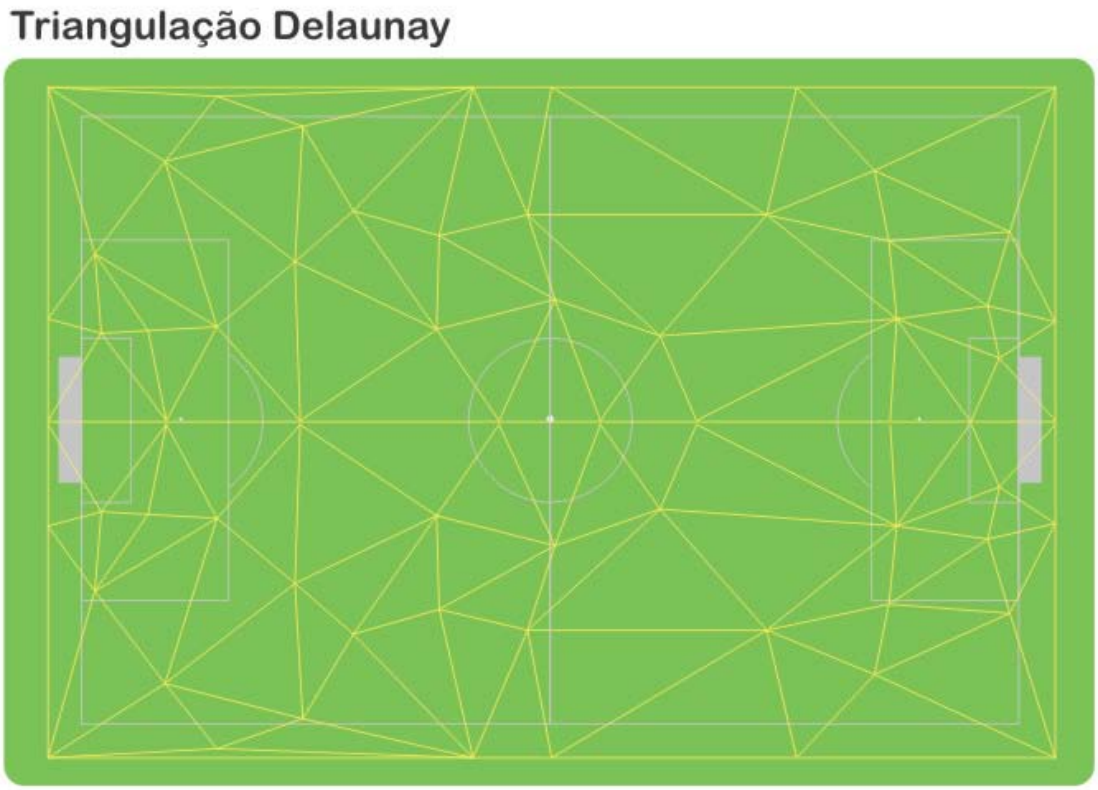


**Figura 18:** Particionamento do campo por Triangulação

Outra sugestão seria melhorar a comunicação entre jogadores, transmitindo mais dados do modelo de mundo e compartilhando melhor a informação entre eles.

Já nas primeira partida, contra o time UvA Trilearn ficou evidente que mesmo tendo implementado a maioria das habilidades avançadas de um jogador a interação social dos agentes ainda é precária. O resultado obtido após a realização de três partidas consecutivas é apresentado na Tabela 16.

**Tabela 16:** Partidas realizadas contra UvA Trilearn

| | UESBots v0.02 | UvA Trilearn v2005 |
|---|---|---|
| **Partida 1** | 0 | 29 |
| **Partida 2** | 0 | 33 |
| **Partida 3** | 1 | 28 |

A sugestão é uma revisão nos módulos básicos de Sensor, Conector e Atuador que apresenta problema de sincronização com o Servidor RoboCup. Através de testes

foi percebidos que os mesmos comandos levam tempos diferentes para serem enviados ao servidor e alguns comandos parecem se perder. Uma das características do Servdor RoboCup é aceitar uma quantidade de comandos em um intervalo delimitado de ciclos. O componente Conector parece não levar isso em consideração, excedendo o limite aceito pelo Servidor.

Uma revisão no código do ModeloMundo também se faz necessária melhorando os métodos de armazenamendo e busca de objetos memorizados. Algumas estruturas de dados podem ser revistas, como por exemplo as classes auxiliares.

No geral a avaliação é positiva, pois houve um avanço no projeto, ampliando o trabalho de RIBEIRO (2010) e desenvolvendo uma ótima base para implementação do componente Raciocínio, melhorando a camada reativa e implementando a base de uma camada cognitiva com métodos preditivos. Outra adição positiva ao projeto foram os métodos de comportamento social baseado em formações táticas e áreas de atuação.

# 5. Conclusão

Nesse capítulo final serão apresentados e analisados os resultados obtidos e também sugestões de temas para trabalhos futuros, bem como as contribuições realizadas por esse trabalho.

Estudantes e pesquisadores de todo o mundo tem despertado interesse nos torneios de futebol de robôs promovidos pela RoboCup. Como um exemplo de Sistemas Multiagentes que é uma área da inteligência artificial distribuída, estas competições e seus desafios científicos incentiva a pesquisa e o aperfeiçoamento de técnicas de IA. Partindo dessas motivações, o presente trabalho teve como objetivos desenvolver um time de futebol de robôs competitivo com componentes que controlem as estratégias e esquemas táticos que o time deve ter em cada momento do jogo.

Tendo em vista que o objetivo principal da implementação do time UESBots de futebol de robôs simulado 2D, foco deste projeto, é o desenvolvimento do componente Raciocínio com processamento reativo e cognitivo com métodos preditivos, além de um comportamento social entre agentes com formação tática e posicionamento em áreas de atuação, os resultados foram alcançados com êxito.

Considerando que com o desenvolvimento deste trabalho houve uma ampliação significativa no âmbito do projeto de implementação de um time competitivo de futebol de robôs para a UESB, torna o UESBots uma base para futuras implementações.

Com o objetivo de dar continuidade no presente trabalho de pesquisa, torna-se necessário revisar o esquema de particionamento do campo para uma divisão mais compartimentada visando uma melhor modelagem das áreas de atuação dos jogadores em seus diversos tipos. Uma revisão nos componentes Sensor, Conector e Atuador se fazem necessários para resolver problemas de sincronização com o Servidor RoboCup. Uma revisão no código do ModeloMundo trará mais dinamismo na atualização e busca de objetos do jogo.

Outros aspectos que podem ser desenvolvidos são:

- Melhorar o protocolo de comunicação entre os agentes de modo a ampliar a visão de mundo de cada um em relação ao ambiente.

- Implementação de uma camada de aprendizagem por de forma a tornar o time capaz de aprender padrões e soluções a medida que vai ampliando a quantidade de jogos realizados.
- Aperfeiçoar interface dos componentes existentes e programar mais habilidades avançadas para os jogadores, dando maior autonomia para jogador completar suas metas.

# 6. Referências

ASIMOV, I. Eu, Robô. Ediouro, São Paulo, SP, 2004 (original 1950).

BARBOSA, L. R. C. Um sistema multiagente para monitoramento atmosférico. 2005. Dissertação (Mestrado em redes de computadores), Universidade Salvador, 2005.

BOTELHO W. T.; Rosa P. F. F. Um sistema de identificação e adaptação pervasivo para a casa inteligente utilizando sistemas multiagentes. Dissertação, Instituto Militar de Engenharia. 2005.

BOER R.; KOK J. The Incremental Development of a Synthetic Multi-Agent System: The UvA Trilearn 2001 Robotic Soccer Simulation Team. Tese (Mestrado em Inteligência Artificial), Faculty of Science University of Amsterdam. 2002.

BÊRNI, D. A. Teoria dos Jogos: Jogos de Estratégia, Estratégia Decisória, Teoria da Decisão. Reichmann and Affonso Ed., Rio de Janeiro, 2004. Disponível em: < http://bit.ly/dO9Rwv>. Acesso em: 01 jan. 2011.

BRAGA, B. T. Da R.; PEREIRA, J. A. Agentes inteligentes - conceitos, características e aplicações. Dissertação (Graduação em Processamento de Dados): Universidade Da Amazônia, 2001. Disponível em: <http://bit.ly/fuvq0s >. Acesso em: 01 jan. 2011.

COSTA, M. T. C. Uma Arquitetura Baseada em Agentes para Suporte ao Ensino à Distância. 1999. Tese (Doutorado em Engenharia de Produção), Universidade Federal de Santa Catarina, Florianópolis, 1999.

DEEP BLUE. Disponível em: < http://bit.ly/e9uIEu>. Acesso em: 01 jan. 2011.

KHASHABI, D. An Introduction to RoboCup and Soccer Simulation 2D, Amirkabir University of Technolgy, 2009.

KITANO, H. RoboCup: The Robot World Cup Initiative. Abertura da I Conferência Internacional de Agentes Autônomos (Agents' 97), Marina del Ray, The ACM Press, 1997.

LÉVY, P. A Inteligência Coletiva. Por Uma Antropologia do Ciberespaço. 4ª Edição, Loyola, São Paulo, 2003.

RIBEIRO, M. S. Prototipação de agentes com arquitetura em camadas e baseada em componentes para desenvolvimento do time de futebol de robôs da UESB. Monografia (graduação Em Ciência da Computação): Universidade Estadual do Sudoeste da Bahia, 2010.

RIOS, A. R. UESBTeam: Utilização de técnicas de Engenharia de Software e Inteligência Artificial para construção de um time de futebol de robôs. 2006. Monografia (Graduação em Ciência da Computação), Universidade Estadual do Sudoeste da Bahia, 2006.

ROBOCUP. A brief History of RoboCup. Disponível em: < http://bit.ly/hd8YfX >. Acesso em: 03 jan. 2011.

ROBOCUP. RoboCup Objective. Disponível em: < http://bit.ly/hKrKtp >. Acesso em: 03 jan. 2011b.

ROBOCUP. What´s RoboCup. Disponível em: < http://bit.ly/e97uX9 >. Acesso em: 03 jan. 2011c.

ROCHA A. Introdução aos agentes inteligentes e aos sistemas multiagentes. Dissertação (Mestrado), Universidade Federal de Lavras - UFLA, Lavras, 2003.

RUSSELL, S., NORVIG, P. Inteligência Artificial, Editora Campus, 2004.

SILVA, A. T. R. Comportamento Social Cooperativo na Realização de Tarefas em Ambientes Dinâmicos e Competitivos. 2006. Dissertação (Mestrado em Sistemas e Computação), Instituto Militar de Engenharia.

SOUZA, E. M. Uma estrutura de agentes para assessoria na internet. 1996. Dissertação (Mestrado em engenharia), Universidade Federal de Santa Catarina, 1996.

SOUSA, M. A. F. et al. Aplicação de um modelo matemática em uma equipe de futebol de robôs. Instituto Militar de Engenharia: Simpósio de Pesquisa Operacional E Logística da Marinha, 2007. Disponível em: <http://bit.ly/gZ8KDm>. Acesso em: 20 dez. 2010.

WOOLDRIDGE, M.; JENNINGS, N. Intelligent Agents: Theory and Practice. 10ª ed. Manchester: Cambridge University Press, 1995.

XAVIER, R.; BARBOSA, R.; MATSUURA, J. O time de futebol simulado ITANDOIRDS-2D. Instituto Tecnológica de Aeronáutica: Anais do XXVI Congresso da SBC, 2006.