

# Agentes e Inteligência Artificial Distribuída

Stats Collector



Universidade do Porto

---

Faculdade de Engenharia

**FEUP**

Nelson Jorge Silva Rodrigues – [nelson@fe.up.pt](mailto:nelson@fe.up.pt)  
Ricardo Jorge Marques Veloso – [ricardo.veloso@fe.up.pt](mailto:ricardo.veloso@fe.up.pt)

Faculdade de Engenharia da Universidade do Porto  
Rua Roberto Frias, s/n, 4200-465 Porto, Portugal

Dezembro 2003

# 1 Índice

<b>1</b>	<b>ÍNDICE</b> .....	<b>1-2</b>
<b>2</b>	<b>OBJECTIVOS</b> .....	<b>2-3</b>
<b>3</b>	<b>DESCRIÇÃO</b> .....	<b>3-4</b>
3.1	FUNCIONALIDADES .....	3-4
3.2	ESTRUTURA DO PROGRAMA .....	3-4
3.2.1	<i>Broker</i> .....	3-5
3.2.2	<i>DataContainer</i> .....	3-5
3.2.3	<i>Agentes Estatística</i> .....	3-5
3.3	ESQUEMAS DE REPRESENTAÇÃO DE CONHECIMENTO E METODOLOGIAS.....	3-5
3.4	DETALHES DE IMPLEMENTAÇÃO .....	3-6
3.4.1	<i>Agente Broker</i> .....	3-6
3.4.1.1	Comunicação com o servidor <i>soccerserver</i> .....	3-6
3.4.1.2	Comunicação com o utilizador.....	3-6
3.4.1.3	Comunicação com os agentes estatística.....	3-7
3.4.2	<i>Classe Agent</i> .....	3-7
3.4.3	<i>Agentes estatística</i> .....	3-7
3.4.3.1	Resultado de jogo.....	3-7
3.4.3.2	Tempo parado de jogo.....	3-7
3.4.3.3	Ocorrências de jogo .....	3-7
3.4.3.4	Acções dos jogadores.....	3-8
3.4.3.5	Posse de bola.....	3-8
3.4.3.6	Posição dos jogadores .....	3-8
3.4.3.7	Perdas de bola .....	3-8
3.4.3.8	Recuperações de bola.....	3-8
3.5	AMBIENTE DE DESENVOLVIMENTO .....	3-8
<b>4</b>	<b>CONCLUSÕES</b> .....	<b>4-9</b>
<b>5</b>	<b>MELHORAMENTOS</b> .....	<b>5-10</b>
<b>6</b>	<b>BIBLIOGRAFIA</b> .....	<b>6-11</b>
<b>7</b>	<b>APÊNDICE</b> .....	<b>7-12</b>
7.1	MANUAL DO UTILIZADOR .....	7-12
7.2	EXEMPLO DE UMA EXECUÇÃO .....	7-12

## 2 Objectivos

Este trabalho tem como objectivo o desenvolvimento de um programa agente, capaz de analisar o comportamento e recolher estatísticas de um jogo de futebol robótico simulado, realizado sobre a plataforma de comunicação *soccerserver*. O programa foi desenvolvido na linguagem de programação C++ e é baseado na plataforma de desenvolvimento Qt.

## 3 Descrição

### 3.1 Funcionalidades

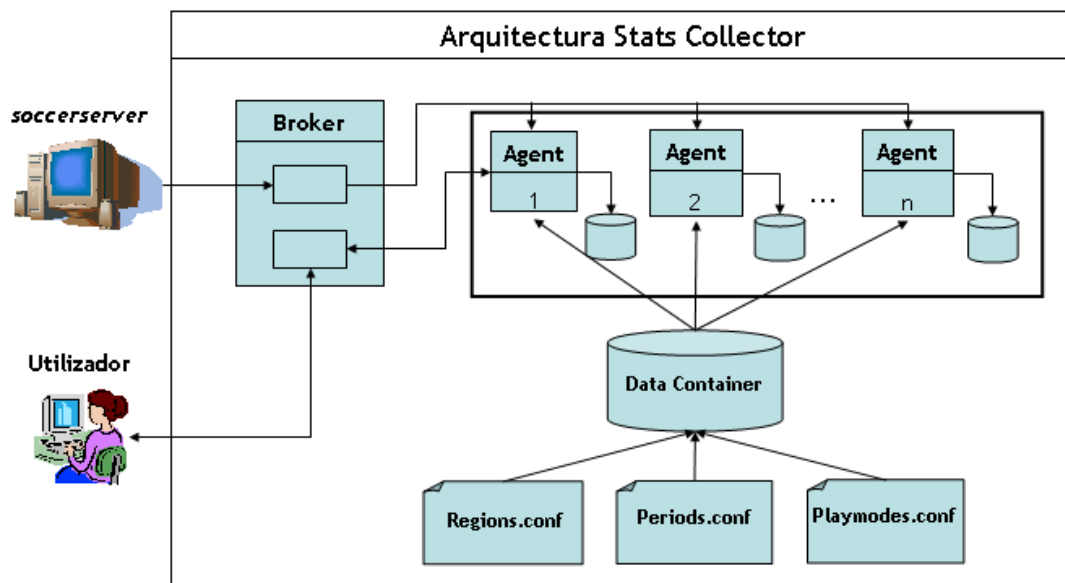
O programa permite a recolha de informação estatística de variados tipos sobre um jogo de futebol robótico simulado, estas estatísticas englobam praticamente todos os aspectos relevantes de um jogo de futebol e que possam ter interesse para uma completa análise da performance de uma equipa. Foram desenvolvidos agentes capazes de recolher o resultado do jogo, tempo parado de jogo, ocorrências de jogo, acções realizadas por um jogador, posse de bola, posição dos jogadores no campo, perdas e recuperações de bola.

Todas as estatísticas são recolhidas tendo em conta a região do campo em que os eventos acontecem e o período de tempo actual. Tanto as regiões de campo como os períodos de tempo são parametrizáveis através de ficheiros de configuração apropriados.

É possível aceder à informação contida pelo agente através de uma interface de linha de comandos utilizando para isso um subset da linguagem COACH UNILANG.

### 3.2 Estrutura do Programa

O programa desenvolvido baseia-se num sistema multi-agente em que sobressaem três grandes módulos: agente *Broker*, objecto *DataContainer* e agentes estatísticas.



### 3.2.1 Broker

Sobre este agente cai a responsabilidade de realizar toda a interface com o exterior do sistema, seja esta feita com o servidor *soccerserver*, ou através da linha de comandos com o utilizador.

Através da comunicação com o servidor é recebida a informação sobre o estado actual da simulação e cabe a este agente a tarefa de trazer esta informação para o domínio do sistema, através da transformação da informação de baixo nível recebida em objectos percebidos pelos restantes intervenientes no sistema.

Pela linha de comandos é estabelecida a interface com o utilizador do programa, cabendo ao agente Broker a responsabilidade de analisar o pedido do utilizador e encaminhá-lo para o agente estatística mais apropriado para o responder.

### 3.2.2 DataContainer

É neste objecto que está concentrada toda a informação relativa à parametrização do sistema.

É constituído por vários dicionários que guardam as características das regiões de campo definidas, dos períodos de tempo, das equipas em jogo e das ocorrências de jogo que interessam recolher.

### 3.2.3 Agentes Estatística

São estes agentes os responsáveis por armazenar a informação recolhida a partir da análise do estado do mundo. Existe um agente para cada estatística recolhida, isto é, existe um agente para recolher o resultado do jogo, um agente para recolher o número de ciclos que cada jogador passa numa região do campo...

É também da responsabilidade do agente produzir uma resposta a pedidos de informação vindos do Broker, esta resposta é depois encaminhada de volta ao utilizador.

## 3.3 *Esquemas de Representação de Conhecimento e Metodologias*

O estado da simulação é guardado em objectos que encapsulam toda a informação relevante.

**Classe Game:** modo de jogo actual, tempo actual, vector de jogadores e bola.

**Classe PlayerJ:** informação sobre a posição e velocidade do jogador, ângulo da cabeça e do corpo, energia do jogador, contagem de remates e apanhos de bola.

**Classe Ball:** informação sobre a posição e velocidade da bola.

As definições da configuração das regiões do terreno e dos períodos de tempo é guardada em dicionários de objectos FieldRegion e TimePeriod.

Para armazenar a informação relativa às estatísticas cada agente possui uma matriz n-dimensional em que cada dimensão representa um dos aspectos que é necessário guardar, isto é, teremos uma dimensão para a região do campo, uma dimensão para o período de tempo...

### **3.4 Detalhes de Implementação**

Seguem-se os detalhes da implementação dos mais importantes componentes do sistema desenvolvido.

#### **3.4.1 Agente Broker**

##### **3.4.1.1 Comunicação com o servidor *soccerserver***

A comunicação entre o agente Broker e o servidor é efectuada através da classe MySocket importada do projecto TeamDesiger. Esta classe faz uso do mecanismo de sinais e slots disponibilizado pela plataforma Qt para informar o agente de que existe nova informação chegada do servidor.

Seguidamente é efectuado o tratamento da informação recebida por forma a transformá-la nos objectos que representam o estado do mundo referidos anteriormente. Esta informação é então enviada, novamente utilizando o sistema de sinais e slots, aos agentes estatística para ser analisada.

##### **3.4.1.2 Comunicação com o utilizador**

A comunicação entre o agente e o utilizador é efectuada utilizando a linha de comandos. Ao ser inicializado e depois de terem sido introduzidas duas equipas na simulação é lançado um thread que executa concorrentemente com a comunicação com o servidor. Cabe a este thread a tarefa de receber os pedidos do utilizador e encaminhá-los para os agentes respectivos. A comunicação com os agentes estatística é efectuada através do chamamento de métodos nos agentes.

Não são tomadas nenhuma precauções para prevenir a existência de race conditions na execução deste thread uma vez que este apenas lê as estruturas partilhadas com a comunicação com o servidor.

### **3.4.1.3 Comunicação com os agentes estatística**

A comunicação com os agentes estatística é efectuada através da utilização do mecanismo de sinais e slots da plataforma Qt. Este é um sistema de eventos que permite criar um evento no agente estatística e nesse evento incluir a informação relativa ao estado do mundo. A arquitectura deste sistema permite adicionar novos agentes estatística sem que para isso seja necessária a alteração do agente Broker, este não sabe que agentes estatística existem.

### **3.4.2 Classe Agent**

Esta é a classe base de todos os agentes estatística e é a classe que permite que todos os agentes tenham a mesma interface perante o Broker. Esta classe implementa também o tratamento do pedido de informação do utilizador, cabendo aos agentes derivados apenas a tarefa de fornecer a informação.

### **3.4.3 Agentes estatística**

Todos os agentes estatística estendem a classe Agent, o que lhes facilita bastante a tarefa de comunicar com o Broker visto que esta já está implementada naquela classe. A cada ciclo de simulação que passa os agentes recebem o novo estado do mundo, guardam o estado anterior e é através da comparação dos dois estados que é possível determinar as diferenças entre estes e conseqüentemente contabilizar os acontecimentos relevantes.

Segue-se uma descrição das estratégias utilizadas por cada agente para a recolha da informação.

#### **3.4.3.1 Resultado de jogo**

Para cada ciclo de jogo verificar se o modo de jogo se alterou, se o novo modo de jogo for um de “goal\_l” ou “goal\_r” aumentar a contagem de golos para a equipa respectiva.

#### **3.4.3.2 Tempo parado de jogo**

Para cada ciclo de jogo se o modo de jogo for diferente de “play\_on” incrementar a contagem de ciclos parados para todos os períodos de tempo que contenham o tempo actual.

#### **3.4.3.3 Ocorrências de jogo**

Para cada ciclo de jogo em que se altere o modo de jogo verificar se este pertence a alguma das ocorrências definidas como interessantes. Caso se verifique esta condição incrementar a contagem de ocorrência para todas as regiões de campo que contenham a posição da bola, e para todos os períodos de tempo que contenham o tempo actual.

#### **3.4.3.4 Acções dos jogadores**

Para cada ciclo de jogo verificar se houve algum jogador que tenha alterado a contagem de toques na bola. Verificar se este toque corresponde a um passe, cruzamento, remate ou simplesmente a controlo de bola. Caso seja passe ou centro determinar o jogador de destino, depois actualizar a contagem da ocorrência tendo em atenção a região de campo de partida, de chegada e o período de tempo actual.

#### **3.4.3.5 Posse de bola**

Para cada ciclo de jogo determinar se algum jogador tem a posse de bola, caso seja verdade actualizar a contagem de ciclos para a região de campo em que o jogador se encontra e para o período de tempo actual.

#### **3.4.3.6 Posição dos jogadores**

Para cada ciclo de jogo e para cada jogador actualizar a contagem de ciclos para a região de terreno em que o jogador se encontra.

#### **3.4.3.7 Perdas de bola**

Para cada ciclo de jogo verificar se a equipa com a posse de bola é diferente da equipa com a posse de bola no ciclo anterior. Caso seja verdade actualizar a contagem de perdas de bola para o período de tempo actual.

#### **3.4.3.8 Recuperações de bola**

Sempre que uma equipa recupera a bola nos 10 ciclos a seguir a tê-la perdido é contabilizada uma recuperação de bola na região de campo que contém a posição da bola no período de tempo actual.

### **3.5 Ambiente de Desenvolvimento**

O trabalho foi desenvolvido numa máquina com processador Intel Pentium III, com 256Mb RAM, sobre o sistema operativo SlackWare Linux versão 9.0.

O programa foi implementado na linguagem de programação C++, utilizando o editor Kate e o compilador g++ versão 3.2.2. Algumas bibliotecas de software foram importadas do projecto TeamDesigner, <object>, <types>, <utils>, <TreatInfo>, <MySocket>. O nosso projecto faz grande uso de bibliotecas disponibilizadas pela plataforma Qt versão 3.1.2, como por exemplo a utilização de *QWidget*, *QObject* e *QString*.

## 4 Conclusões

O estudo da plataforma de simulação *soccerserver* e a aprendizagem de uma nova linguagem de programação, C++, estendeu-se por um longo período de tempo. Esta extensão causou um encurtamento no tempo de desenvolvimento.

Contudo foi um trabalho interessante de realizar devido à área em que se enquadra. Futuramente, com mais tempo dedicado a melhoramentos, poderia resultar num módulo bastante útil e importante numa equipa de futebol robótico simulado.

## 5 Melhoramentos

As estatísticas das acções são apenas contabilizadas e o seu resultado não distinguível. Por exemplo um remate não é identificado como bem ou mal direccionado. Poderíamos neste caso, com mais tempo, aperfeiçoar esta funcionalidade de modo a colmatar esta limitação.

A visualização de informação em modo gráfico seria um importante melhoramento a implementar, o que possibilitaria uma melhor análise das estatísticas tratadas pelos agentes. A ter em conta que, a representação em modo gráfico de algumas estatísticas não é viável devido à enorme quantidade de informação.

O programa apenas se consegue ligar a servidores *soccerserver* que estejam a correr na mesma máquina, sendo esta uma grande limitação do sistema desenvolvido.

## 6 Bibliografia

- Tese de doutoramento do Prof. Luís Paulo Reis
- RoboCup SoccerServer User Manual
- Relatório Final de Projecto “FCPortugal Uma equipa da liga de simulação do RoboCup” – Cláudio Teixeira e Johnny Santos

## 7 Apêndice

### 7.1 Manual do Utilizador

Depois de ter iniciado uma instância do servidor *soccerserver*, executar o programa numa linha de comandos utilizando o comando *collector*. Depois de terem entrado as duas equipas na simulação é apresentado ao utilizador um prompt onde este pode inserir comandos para serem executados pelo agente.

As frases reconhecidas são:

```
(game_result <PERIOD>)  
(stopped_time <PERIOD>)  
(game_occurrence <GAME_PLAYMODE> <REGION> <TEAM> <PERIOD>)  
(action <ACTION> <REGION_FROM> <REGION_TO> <TEAM> <PLAYER> <PERIOD>)  
(ball_possession <REGION> <TEAM> <PLAYER> <PERIOD>)  
(player_position <REGION> <TEAM> <PLAYER> <PERIOD>)  
(ball_losses <ACTION> <REGION > <TEAM> <PLAYER> <PERIOD>)  
(ball_recoveries <RECOVERY> <REGION > <TEAM> <PLAYER> <PERIOD>)
```

Sempre que o utilizador insira uma frase não completa o agente responde com todas as possibilidades de preenchimento do argumento não inserido, por exemplo:

```
$ (game_result)  
(game_result game 3 2 )  
(game_result first_half 1 0 )  
(game_result second_half 2 2 )  
(game_result extra_time 0 0 )
```

### 7.2 Exemplo de uma Execução

Seguidamente é apresentada um screen shot de uma execução sobre uma simulação em que participaram as equipas FCPortugal2002 e TsinghuAeolus. O utilizador fez vários pedidos de informação, o resultado de jogo, o número de offsides da equipa TsinghuAeolus na primeira parte, um pedido de informação mal formado, o número de perdas de bola da equipa FCPortugal2002 no jogo todo no terreno de jogo e o número de remates efectuados pela equipa TsinghuAeolus durante o jogo todo.

```
xterm
$ (game_result)
(game_result last_500 0 0 )
(game_result first_half 0 5 )
(game_result game 2 8 )
(game_result last_1000 0 2 )
(game_result last_100 0 0 )
(game_result extra_time 0 0 )
(game_result last_20 0 0 )
(game_result second_half 2 3 )
(game_result last_50 0 0 )
$ (game_occurrence offside TsinghuAeolus first_half)
(game_occurrence offside left_penalty_box TsinghuAeolus first_half 0 )
(game_occurrence offside upper_wing TsinghuAeolus first_half 0 )
(game_occurrence offside field TsinghuAeolus first_half 1 )
(game_occurrence offside right_middle_field TsinghuAeolus first_half 0 )
(game_occurrence offside right_penalty_box TsinghuAeolus first_half 0 )
(game_occurrence offside left_middle_field TsinghuAeolus first_half 1 )
(game_occurrence offside lower_wing TsinghuAeolus first_half 0 )
$ (ball_losses FCPortugal game field)
$ (ball_losses FCPortugal2002 game field)
Query mal formada.
$ (ball_losses FCPortugal2002 game field)
(ball_losses field FCPortugal2002 1 game 2 )
(ball_losses field FCPortugal2002 2 game 1 )
(ball_losses field FCPortugal2002 3 game 5 )
(ball_losses field FCPortugal2002 4 game 8 )
(ball_losses field FCPortugal2002 5 game 3 )
(ball_losses field FCPortugal2002 6 game 11 )
(ball_losses field FCPortugal2002 7 game 10 )
(ball_losses field FCPortugal2002 8 game 22 )
(ball_losses field FCPortugal2002 9 game 25 )
(ball_losses field FCPortugal2002 10 game 8 )
(ball_losses field FCPortugal2002 11 game 4 )
$ (action remate TsinghuAeolus game field->field)
(action remate field->field TsinghuAeolus 1 game 0 )
(action remate field->field TsinghuAeolus 2 game 0 )
(action remate field->field TsinghuAeolus 3 game 1 )
(action remate field->field TsinghuAeolus 4 game 1 )
(action remate field->field TsinghuAeolus 5 game 2 )
(action remate field->field TsinghuAeolus 6 game 1 )
(action remate field->field TsinghuAeolus 7 game 1 )
(action remate field->field TsinghuAeolus 8 game 3 )
(action remate field->field TsinghuAeolus 9 game 3 )
(action remate field->field TsinghuAeolus 10 game 5 )
(action remate field->field TsinghuAeolus 11 game 3 )
$ █
```