

UNIVERSIDADE DO MINHO

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA

PERFIL **Sistemas Inteligentes**

UNIDADE CURRICULAR DE **Agentes Inteligentes**

EDIÇÃO 2015/2016

---

## AGENTES E SISTEMAS MULTIAGENTE

---

AUTORES:

Diana Oliveira (a67652)

Gil Gonçalves (a67738)

Pedro Duarte (a61071)

Pedro Lima (a61061)

Braga, 8 de Janeiro de 2017





## **Resumo**

No presente documento iremos fazer um estado da arte mais teórico sobre os agentes e sua aplicação a domínios concretos com o objetivo de conhecer os principais conceitos da computação baseada em Agentes e identificar e caracterizar diferentes áreas de aplicação.

Também será descrita uma componente aplicacional mais prática onde estará concebida uma arquitetura distribuída baseada em agentes para um dado problema escolhido pelo grupo que permitiu conhecer as principais ferramentas oferecidas pela plataforma Jade.

# Conteúdo

0.1	Introdução . . . . .	3
<b>1</b>	<b>Estado da arte sobre agentes</b>	<b>4</b>
1.1	Entendendo Agentes . . . . .	4
1.2	Aplicações em Domínios . . . . .	5
1.3	Propriedades e Vertentes . . . . .	6
1.4	Agentes vs Objetos . . . . .	7
1.5	Agentes vs Sistemas Periciais . . . . .	7
1.6	Tipos de Agentes . . . . .	7
1.7	Arquiteturas para Agentes de Software . . . . .	8
1.8	Sistemas Multiagentes . . . . .	10
1.8.1	Vantagens de um sistema multiagente . . . . .	10
1.8.2	Comunicação entre agentes . . . . .	10
1.8.3	Linguagens de Comunicação entre Agentes de Software . . . . .	10
1.8.4	Arquiteturas em sistemas multiagente . . . . .	11
1.8.5	Cooperação em sistemas multiagente . . . . .	12
1.8.6	Sistemas baseados em Quadros Negros . . . . .	12
1.8.7	Sistemas baseados em troca de mensagens . . . . .	13
1.8.8	Coordenação em sistemas multiagente . . . . .	13
<b>2</b>	<b>Desenvolvimento de uma Arquitetura Distribuída baseada em Agentes</b>	<b>14</b>
2.1	Problema escolhido . . . . .	14
2.2	Construção da Base de Conhecimento . . . . .	15
2.2.1	Fontes e respectiva extração de dados . . . . .	15
2.3	Critérios de decisão . . . . .	24
2.4	Desenvolvimento da solução . . . . .	25
2.4.1	Arquitetura . . . . .	25
2.5	Descrição de Agentes - Modelo Físico . . . . .	26
2.5.1	Agentes <i>Man1</i> e <i>PJ</i> . . . . .	26
2.5.2	Agente <i>Critic</i> (Crítico) . . . . .	27
2.5.3	Agente <i>AE</i> . . . . .	27
2.5.4	Agente <i>AJ</i> . . . . .	28
2.5.5	Agente <i>AH</i> . . . . .	28
2.5.6	Agente <i>ACL</i> . . . . .	29

2.5.7	Agente <i>Software</i> . . . . .	29
2.6	Apresentação de Resultados . . . . .	29
2.6.1	Funcionamento do programa . . . . .	30
2.6.2	Preparação do programa . . . . .	32
2.6.3	Correr o programa . . . . .	32
2.7	Conclusão e trabalho futuro . . . . .	33
2.8	Bibliografia . . . . .	34

## 0.1 Introdução

Um agente é algo que age, capaz de produzir um efeito. É algo capaz de perceber o seu ambiente por meio de sensores e de agir sobre esse ambiente através de atuadores. O agente deve tomar a ação "correta" baseado no que ele percebe para ter sucesso.

Um agente autônomo possui algum conhecimento inicial e a habilidade de inferir ou aprender novos conhecimentos. O comportamento do agente pode depender de dois fatores: do conhecimento embutido no programa e da sua própria experiência.

Num sistema multi-agente onde existem vários agentes a trabalhar é necessário que existam mecanismo de comunicação e como esses agentes podem estar em vários sistemas diferentes foi necessário definir protocolos de comunicação para que um agente soubesse comunicar com outros agentes que tivessem em diferentes máquinas.

# Capítulo 1

## Estado da arte sobre agentes

### 1.1 Entendendo Agentes

Um agente num sentido alegórico foi introduzido no âmbito da comunidade de Inteligência Artificial e pode ser usado em várias área do conhecimento humano como a Psicologia, Economia, Sociologia assim como em ciências da computação envolvendo a área do presente trabalho.

Agentes são entidades a quem se delegam, por parte de humanos, tarefas para executar tirando partido de capacidades de cálculo, memorização e/ou perseverança. São portanto componentes persistentes e ativos capazes de produzir um efeito através do modo como percebem e raciocinam a informação que obtém, de um ambiente complexo, inserido através de sensores, comunicando e atuando, executando um conjunto de tarefas, nesse mesmo ambiente através de atuadores.

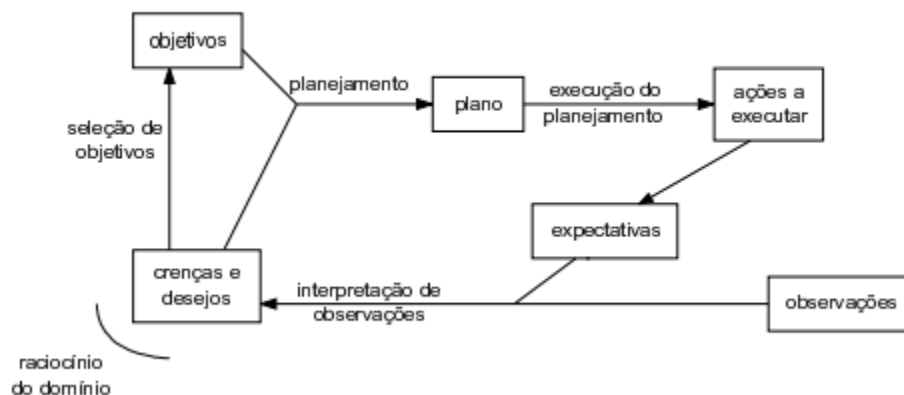


Figura 1.1: Um modelo de agente

## 1.2 Aplicações em Domínios

- Na área industrial:
  - Controlo de processos;
  - Produção;
  - Controlo/Simulação de sistemas de controlo de Tráfego Aéreo.
- No sector da agricultura
  - Automatização de trabalhos agrícolas: utilização de sensores e aprendizagem máquina para monitorizar constantemente o ambiente e o crescimento das plantas e ajustar a iluminação, a temperatura, a humidade, a água e nutrientes do solo para maximizar a produtividade;
  - Antecipação de doenças nas plantações: utilização de sensores e um algoritmo de aprendizagem máquina para monitorizar as colheitas e alertar logo que uma planta esteja doente. O algoritmo pode identificar sinais de bolor, bactérias, danos causados por insetos, bem como analisar dados de nutrição.
- No sector Empresarial
- No sector do Consumo
- Previsão de Desastres e meio ambiente
- Educação
- Segurança Pública
- No sector Comercial:
  - Comércio Eletrónico;
  - Gestão de Empresas.
- No ensino (e.g., em sistemas de *e-learning*).
- No domínio da Medicina:
  - Monitorização de pacientes;
  - Cuidados de Saúde;
  - Melhoria da acessibilidade a pessoas com deficiências físicas: Uma iniciativa de investigação financiada pela União Europeia chamada *Robotic Adaption to Humans Adapting to Robots (Radhar)* desenvolveu uma cadeira de rodas robótica que utiliza algoritmos de visão computacional para ajudar os usuários a percorrer melhor o ambiente. A cadeira de rodas de *Radhar* faz *scan* ao ambiente 10 vezes por segundo para mapear os obstáculos e gerar rotas potenciais que a cadeira de rodas poderia tomar.



- Entretimento:
  - Jogos.

De notar que estas áreas acabadas de referir, têm alguns denominadores comuns, tais como:

- A distribuição geográfica, a temporalidade, um certo conteúdo semântico ou mesmo funcional;
- A complexidade.

## 1.3 Propriedades e Vertentes

Um agente ou sistema multi-agente diz-se fraco quando não apresenta um conjunto mínimos de atributos como os que listamos de seguida:

- **Autonomia**  
Têm controlo sobre as próprias ações e conhecimento e podem operar sem intervenção de outros agentes
- **Reatividade**  
Aquando mudanças num ambiente em que um agente está inserido, este responde atempadamente e de forma adequada a essas alterações.
- **Pro-atividade**  
São capazes de tomar iniciativa realizando ações correspondentes a um comportamento.
- **Sociabilidade**  
Ao serem apresentados certos problemas, os agentes podem comunicar, cooperar ou competir com outros agentes para a sua resolução.

Um agente ou sistema multi-agente diz-se forte quando apresenta um conjunto de mais-valias como a percetibilidade, a sentimentalidade e o emocionismo. Porém terão de apresentar também um conjunto mínimo de atributos como os que listamos de seguida:

- **Mobilidade**  
Capacidade de um agente se poder movimentar de uma lado para outro.
- **Intencionalidade**  
Capacidade de um agente definir os objetivos e as estratégias para os atingir.
- **Aprendizagem**  
Capacidades de aprendizagem que fazem com que agente adquira novo conhecimento e altere o seu comportamento baseado na sua experiência prévia.

- **Competência**

Um agente é competente quando conduz com sucesso as tarefas de que é responsável.

- **Veracidade**

Assunção de que um agente não comunica informação falsa de propósito.

- **Racionalidade**

Um agente agirá de forma a atingir os seus objetivos e não agirá de forma a impedir que esses mesmos objetivos sejam atingidos.

- **Benevolência**

Os agentes não devem assumir um comportamento contra produtivo e devem sempre tentar fazer aquilo que lhes é solicitado.

- **Emotividade**

Certas características do ser humano começaram a fazer parte dos agentes.

## 1.4 Agentes vs Objetos

O conceito de Objeto consiste numa entidade computacional que encapsula um estado e é capaz de executar um conjunto de ações sobre este estado e comunicar através de mensagens. Embora os objetos possuam autonomia sobre o seu estado não possuem autonomia sobre o seu comportamento. Pelo contrário, os agentes possuem controlo sobre o seu comportamento e, como tal, um outro agente terá de solicitar ao agente que execute uma dada ação.

## 1.5 Agentes vs Sistemas Periciais

Um Sistema Pericial é um programa de computador que utiliza conhecimento específico do domínio de um problema e emula a metodologia e desempenho de um especialista no domínio desse problema. Estes assistem os peritos numa determinada área de conhecimento e executam tarefas de alto nível. No entanto, não têm autonomia, ou seja, executam sempre a mesma ação.

## 1.6 Tipos de Agentes

As características de um agente dependem do tipo de aplicação a que se propõem. Assim sendo, estes são divididos em várias tipologias:

- **Mobilidade**

Agentes estáticos ou móveis. Os agentes móveis, podem estar residentes na sua máquina de origem ou temporariamente numa outra máquina;

- **Modelo de Raciocínio**

Presença ou não de um modelo de raciocínio simbólico, ou seja, um agente pode ser deliberativo ou puramente reativo;

- **Função do Agente**

A função principal assumida pelo agente, como por exemplo um agente de pesquisa de informação ou de interface;

- **Autonomia**

Grau de autonomia do agente;

- **Cooperação**

Realização ou não de ações cooperativas com outros agentes;

- **Aprendizagem**

Inclusão ou não de capacidades de aprendizagem no agente;

- **Características híbridas**

Estas combinam duas ou mais filosofias diferentes num mesmo agente.

## 1.7 Arquiteturas para Agentes de Software

Para definir arquiteturas internas dos agente é necessário, primeiramente, saber qual o tipo de agente que se está a tratar.

A arquitetura de um agente mostra como ele está implementado em relação as suas propriedades, a sua estrutura e como os módulos que a compõem podem interagir, garantindo a sua funcionalidade. Em suma, a arquitetura de um agente especifica a sua estrutura e funcionamento.

Existem vários tipos de arquiteturas para agentes de software entre as quais se destacam:

- **Arquitetura Deliberativa**

Possui um modelo simbólico do mundo ou universo de discurso;

As decisões são tomadas via raciocínio lógico;

Possui um conjunto de metas e intenções, onde é elaborado um plano baseando-se nessas metas;

Possui certas restrições sobre a construção do modelo simbólico.

- **Arquitetura Reativa**

Sem modelo simbólico interno do mundo ou universo de discurso;

Um agente comporta-se como um autómato inserido no meio ambiente que o rodeia;

Procura lidar com a perceção que tem do mundo através da receção de itens de informação do tipo atómico, que lhe são passadas através de sensores;

As decisões tomadas são implementadas em alguma forma de mapeamento direto da situação para a ação, usando regras de condição/ação.

- **Arquitetura Híbrida**

Os agentes reativos são de certa forma redutores, o que dificulta a implementação de estratégias direcionadas para atingir um certo objetivo;

Por outro lado, os agentes deliberativos ao apresentarem formas de raciocínio e/ou de representação de conhecimento porventura complexas, tornam-se, não raras vezes, incapazes de reagir de uma forma rápida a estímulos vindos do exterior;

A ideia principal passa por categorizar as funcionalidades dos agentes em camadas dispostas hierarquicamente, onde à camada reativa é atribuída alguma forma de prioridade sobre a deliberativa, de tal modo que se aproveite uma das suas características mais importantes, que é o de dar uma resposta rápida a eventos detetados no ambiente ou universo de discurso.

- **Arquitetura *BDI* – *Beliefs, Desires, Intentions***

Agentes de crença-desejo-intenção a decisão a tomar depende da manipulação de estruturas de dados representando as crenças, desejos e intenções do agente;

O agente decide, momento a momento, qual ação tomar na procura do seu objetivo, além de envolver dois processos importantes: a decisão de quais os objetivos a alcançar e como irá alcançar esses objetivos.

- **Crenças**

A aceitação de que uma afirmação é verdadeira ou então de que algo existe;

- **Desejos**

Os desejos são derivados das crenças, contém o julgamento dos agentes com relação a situações futuras;

Os desejos são vistos como objetivos e prioridades geradas instantaneamente, ou funcionalmente, mas que não requerem representações de estado.

- **Intenções**

As intenções dos agentes são um subconjunto dos desejos. Se um agente decide seguir um objetivo específico, então este objetivo torna-se uma intenção. A falta de recursos faz com que os agentes não possam seguir todos os objetivos ao mesmo tempo;

Dessa forma, deve haver prioridades para objetivos importantes e processá-las de acordo com seus pesos.

## 1.8 Sistemas Multiagentes

Sistemas Multiagentes formam uma subárea da Inteligência Artificial Distribuída que estuda o comportamento de um conjunto de agentes autônomos objetivando a solução de um problema que está além das suas capacidades individuais.

Um sistema multiagente pressupõe coordenação entre um conjunto existente de agentes autônomos e inteligentes. Fundamentalmente, está envolvida na procura por uma funcionalidade que permita que estes agentes possam coordenar os seus conhecimentos, objetivos, habilidades e planos individuais de uma forma conjunta, para a resolução de um problema.

### 1.8.1 Vantagens de um sistema multiagente

- Maior rapidez na resolução de problemas através do aproveitamento do paralelismo;
- Diminuição da comunicação por transmitir soluções parciais para outros agentes invés de dados brutos para um *data center*;
- Maior flexibilidade por ter agentes de diferentes habilidades dinamicamente agrupados para a resolução de problemas;
- Aumento da segurança por permitir que agentes assumam responsabilidades dos agentes que falharam.

### 1.8.2 Comunicação entre agentes

A comunicação é uma importante característica que os agentes inteligentes possuem para atingirem seus objetivos.

Para a definição de uma linguagem de comunicação entre agentes foram analisados diversos níveis aos quais os sistemas baseados em agentes deve conter:

- Transporte: como os agentes enviam e recebem mensagens;
- Linguagem: qual o significado de mensagens individuais;
- Política: como os agentes estruturam conversações;
- Arquitetura: como conectar sistemas em concordância com protocolos existentes.

### 1.8.3 Linguagens de Comunicação entre Agentes de Software

A Linguagem de Comunicação entre Agentes (*ACL - Agent Communication Language*) foi definida para a comunicação entre agentes, e pode ser dividida em:

- Vocabulário: grande dicionário de palavras apropriadas às áreas de aplicação comum. Também conhecido como ontologia.

A ontologia fornece o vocabulário de representação para o domínio em questão, e

um conjunto de definições e axiomas que restringem o significado dos termos nesse vocabulário, de forma a permitir uma interpretação consistente e única;

- Linguagem interna: *KIF* (*Knowledge Interchange Format*) é uma linguagem do tipo declarativo e suporta uma interpretação de expressões formuladas em termos da Lógica Predicativa de Primeira Ordem;
- Linguagem externa: *KQML* (*Knowledge Query and Manipulation Language*) é uma camada linguística capaz de encapsular estruturas *KIF*, fornecendo uma comunicação mais eficiente.

### 1.8.4 Arquiteturas em sistemas multiagente

A arquitetura de um sistema multiagente mostra a maneira como o sistema está implementado em termos de propriedades e estruturas e como os agentes que o compõem podem interagir a fim de garantir a funcionalidade do sistema.

As arquiteturas dos sistemas multiagente podem ser classificadas de acordo com as necessidades da aplicação, dos usuários e do grau de sofisticação ou nível de inteligência dos agentes.

As arquiteturas de um sistema multiagente podem ser classificadas em três grupos:

- **Arquitetura simples:** quando é composta por um único e simples agente;
- **Arquitetura moderada:** quando é composta por agentes que realizam as mesmas tarefas, mas possuem diferentes usuários e podem residir em máquinas diferentes;
- **Arquitetura complexa:** quando é composta por diferentes tipos de agentes, cada um com uma certa autonomia, podendo cooperar e podendo estar em diferentes plataformas.

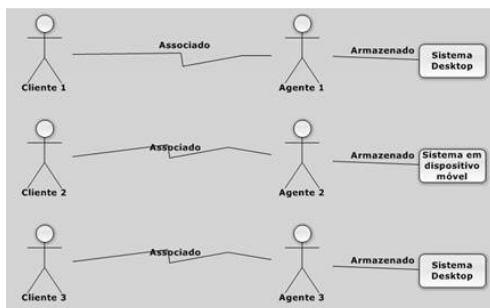


Figura 1.2: Arquitetura moderada

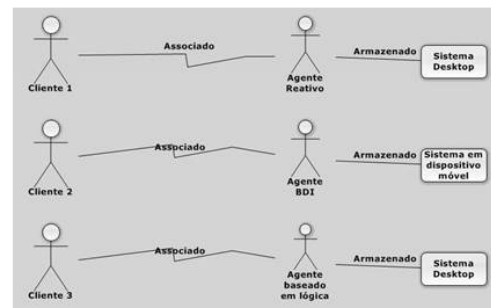


Figura 1.3: Arquitetura complexa

### 1.8.5 Cooperação em sistemas multiagente

Dentro de um sistema multiagente, uma política de cooperação se é necessária, uma vez que é através deste mecanismo que os agentes expressam as suas necessidades a outros agentes a fim de realizar uma determinada tarefa. O mecanismo de cooperação de agentes visa determinar a maneira como os agentes expõem as suas necessidades a outros agentes para atingirem um determinados objetivos.

O processo de cooperação pode acontecer de duas maneiras: Partilha de Tarefas *Task Sharing* e Partilha de Resultados *Result Sharing*. A primeira caracteriza-se pela necessidade dos agentes auxiliar outros agentes durante a execução de uma tarefa, enquanto que na segunda, os agentes disponibilizam informações para a sociedade, prevendo que algum agente possa necessitar dela num determinado momento.

### 1.8.6 Sistemas baseados em Quadros Negros

Um Sistema Baseado em Quadros Negros configura uma espécie de meta-arquitetura, isto é, uma arquitetura a partir da qual é possível gerar novas arquiteturas. Os sistemas que correm nessas arquiteturas afirmam-se por apresentarem formas de organização, estruturação, e utilização do conhecimento deveras eficientes.

Os Agentes não comunicam de maneira direta, mas através de um quadro-negro. O quadro-negro é uma estrutura de dados dividida em regiões ou níveis para facilitar a procura de informação. Funciona como um repositório de mensagens de perguntas e respostas, onde um Agente coloca pergunta e espera que outro Agente processe esta mensagem, para lhe fornecer uma resposta. Estas mensagens podem ser interpretadas como mensagens em que os Agentes solicitam recursos a outros Agentes.

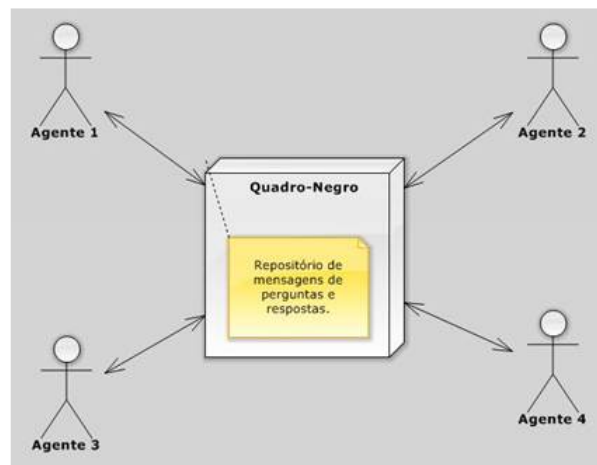


Figura 1.4: : Arquitetura em Quadro Negro

### 1.8.7 Sistemas baseados em troca de mensagens

Os Agentes se comunicam diretamente por troca de mensagens assíncronas. Nesta arquitetura os Agentes precisam saber todos os nomes e endereços de todos os Agentes presentes no sistema. Estas mensagens precisam de obedecer a um protocolo para que possam ser encaminhadas e compreendidas pelos Agentes.

### 1.8.8 Coordenação em sistemas multiagente

A coordenação entre agentes refere-se à maneira como os agentes estão organizados a fim de alcançar um objetivo comum.

Existem dois tipos de ambientes onde os agentes podem estar inseridos:

**Ambientes cooperativo** – passa por um processo de tomada de decisão em que as partes envolvidas negociam, em termos de alcançarem um ou mais objetivos;

**Ambientes competitivos** – passa por um processo de decisão em que as partes envolvidas competem tendo em conta um único objetivo.



## Capítulo 2

# Desenvolvimento de uma Arquitetura Distribuída baseada em Agentes

### 2.1 Problema escolhido

Para o tema do nosso trabalho escolhemos um **sistema de previsões de resultados em jogos de futebol** com a finalidade de ajudar a efetuar apostas com algum fundamento.

Hoje em dia a informação está dispersa por vários websites, então a nossa ideia é recolher informações de várias fontes para automaticamente e com base em critérios relevantes na área sugerir resultados para jogos futuros. O seguinte esquema, apresentado na primeira abordagem do grupo ao tema, pretende explicar de forma resumida a ideia:

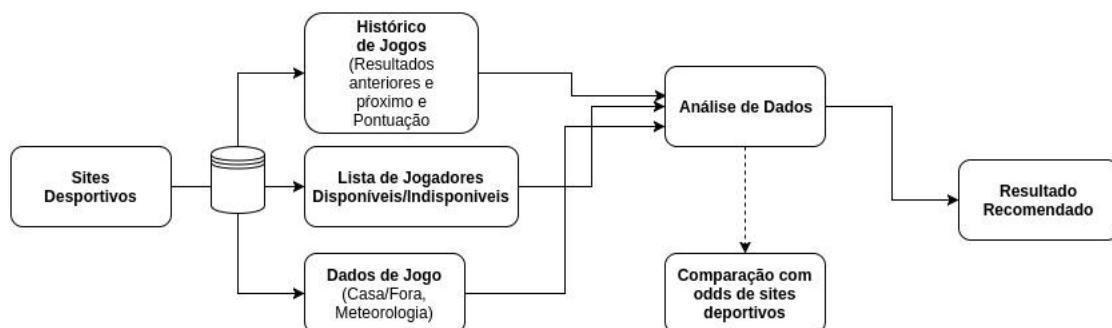


Figura 2.1: Esquema resumo

## 2.2 Construção da Base de Conhecimento

Apesar de não ser o objetivo principal do trabalho, a obtenção da base de conhecimento foi um desafio para o grupo. O facto de não termos encontrado nenhuma fonte onde estivessem presentes todos os dados que necessitaríamos de forma organizada e completa levou-nos à construção de uma base de conhecimento num **formato XML**.

### 2.2.1 Fontes e respectiva extração de dados

- <http://soccerwiki.com/>

Neste website, foi possível extrair conhecimento como as equipas (nome) e jogadores (nome, idade, posição e valor).

Todos os clubes				
Escudo	Clube	Treinador	Estádio	Localidade
	Boavista FC	Erwin Sánchez	Estádio do Bessa	Porto
	CD Feirense	José Mota	Marcolino de Castro	Santa Maria da Feira
	CD Nacional	Manuel Machado	Estádio da Madeira	Funchal
	CD Tondela	Armando Petit	João Cardoso	Tondela
	CF Belenenses	Quim Machado	Estádio do Restelo	Lisboa
	CS Marítimo	Desconhecido	Estádio dos Barreiros	Funchal
	FC Arouca	Lito Vidigal	Municipal de Arouca	Arouca
	FC Porto	Nuno Espírito Santo	Estádio do Dragão	Porto
	GD Chaves	Jorge Simão	Municipal Eng. Manuel Branco Teixeira	Chaves
	GD Estoril Praia	Fabiano Soares	António Coimbra da Mota	Estoril
	Moreirense FC	Augusto Inácio	Comendador Joaquim de Almeida Freitas	Moreira de Cónegos
	Paços de Ferreira	Desconhecido	Estádio Capital do Móvel	Paços de Ferreira
	Rio Ave	Nuno Capucho	Estádio dos Arcos	Vila do Conde
	SL Benfica	Rui Vitória	Estádio da Luz	Lisbon
	Sporting CP	Jorge Jesus	Estádio José Alvalade	Lisbon
	Sporting de Braga	Desconhecido	Estádio Municipal de Braga	Braga
	Vitória de Guimarães	Pedro Martins	D. Afonso Henriques	Guimarães
	Vitória de Setúbal	José Couceiro	Estádio do Bonfim	Setúbal

Figura 2.2: Lista de equipas (vista através de um browser)




























Dados	Nac	Jogador	Pos	Idade	Cl
-		SILVA, Miguel	GR	21	83
-		JESUS, Douglas	GR	33	83
Emp.		GEORGEMY, Gonçalves	GR	21	75
-		FERREIRA, Rúben	D,MD(E)	26	85
-		ALEX, Freitas	D,MD,M,MO,A(E)	25	83
-		GASPAR, Bruno	D(DE),MD(D)	22	83
-		HENRIQUE, Pedro	D(EC)	24	82
-		SÁ, Josué	D,MD(C)	24	85
-		GOUANO, Prince	D,MD(C)	23	83
-		MORENO, João	D,MD(C)	35	80
-		AURELIO, João	D,MD,M,MO(D)	28	85
-		MIRANDA, Rafael	MD,M(C)	32	84
-		ZUNGU, Bongani	MD,M,MO(C)	24	80
-		MBEMBA, Nolan	MD,M(C)	21	77
Emp.		MENSAH, Bernard	M,MO(C)	22	83
-		PHETE, Thibang	M(C)	22	78
-		PEDRO, João	M(C)	23	78
Emp.		MAREGA, Moussa	MO(DE),A(DEC)	25	85
-		TOZÉ, Pinheiro	MO(DEC),A(DE)	23	85
Emp.		HERNÂNI, Fortes	MO,A(DE)	25	84
Emp.		HURTADO, Paolo	MO(DEC),A(DE)	26	83
-		RAPHINHA, Belloli	MO,A(DE)	20	78
-		SILVA, Alexandre	MO,A(DE)	19	78
-		TEXEIRA, David	A(DEC)	25	83
-		SOARES, Tiquinho	A(C)	25	83
-		MENDES, Bruno	A(C)	22	80
-		VIGÁRIO, João	A(C)	21	78

Figura 2.3: Lista de Jogadores (vista através de um browser)

Para extrair a informação do website, foi utilizado primeiramente um método simples como o *wget* mas originava um ficheiro *html* muito difícil de tratar (muito "lixo").

Teve-se assim que arranjar um método mais complexo que só extraí-se a informação importante. Esta foi conseguida em dois passos:

- 1) Usando um **Web Crawler** em **NodeJS**
- 2) Usando uma **script Perl** com recurso a **expressões regulares** para tratar e gerar um **ficheiro XML** com a informação.

```

1  - Miguel SILVA
2
3  SILVA, Miguel GR 21 83
4  - Douglas JESUS
5
6  JESUS, Douglas GR 33 83
7  Emp.
8  Gonçalves GEORGEMY
9
10 GEORGEMY, Gonçalves GR 21 75
11 - Rúben FERREIRA
12
13 FERREIRA, Rúben D,MD(E) 26 85
14 - Freitas ALEX
15
16 ALEX, Freitas D,MD,M,MO,A(E) 25 83
17 - Bruno GASPAR
18
19 GASPAR, Bruno D(DE),MD(D) 22 83
20 - Pedro HENRIQUE
21
22 HENRIQUE, Pedro D(EC) 24 82
23 - Josué SÁ
24
25 SÁ, Josué D,MD(C) 24 85
26 - Prince GOUANO
27
28 GOUANO, Prince D,MD(C) 23 83
29 - João MORENO

```

Figura 2.4: Informação de uma equipa após aplicação de um Web Crawler em NodeJS

```

2  <equipa>
3  <nome_completo>Vitoria de Guimaraes</nome_completo>
4  <jogadores>
5  <jogador>
6  <nome>SILVA Miguel</nome>
7  <posicao>GR</posicao>
8  <idade>21</idade>
9  <valor>83</valor>
10 <status>1</status>
11 </jogador>
12 <jogador>
13 <nome>JESUS Douglas</nome>
14 <posicao>GR</posicao>
15 <idade>33</idade>
16 <valor>83</valor>
17 <status>1</status>
18 </jogador>
19 <jogador>
20 <nome>GEORGEMY Goncalves</nome>
21 <posicao>GR</posicao>
22 <idade>21</idade>
23 <valor>75</valor>
24 <status>1</status>
25 </jogador>
26 <jogador>
27 <nome>FERREIRA Ruben</nome>
28 <posicao>DEF</posicao>
29 <idade>26</idade>
30 <valor>85</valor>
31 <status>1</status>
32 </jogador>
33 <jogador>
34 <nome>ALEX Freitas</nome>
35 <posicao>DEF</posicao>
36 <idade>25</idade>
37 <valor>83</valor>
38 <status>1</status>
39 </jogador>
40 <jogador>
41 <nome>GASPAR Bruno</nome>
42 <posicao>DEF</posicao>
43 <idade>22</idade>
44 <valor>83</valor>
45 <status>1</status>
46 </jogador>

```

Figura 2.5: Informação de uma equipa após a aplicação do script em Perl com REgex

De notar a generalização no campo da posição:  
 Por exemplo "D(DE),MD(D)" foi traduzido para **DEF**, onde terá um factor importante nos critérios de decisão que em seguida falaremos como também em trabalho futuro.

*<http://www.zerozero.pt/>*

Neste website foi possível obter uma lista de jogos passados e futuros de uma determinada liga.

<a href="#">&lt; anterior</a>	JORNADA 12			<a href="#">seguite &gt;</a>
02/12	<b>Marítimo</b>	2-1	Benfica	 
03/12	<b>Rio Ave</b>	3-1	Tondela	 
	<b>Sporting</b>	2-0	V. Setúbal	 
	<b>FC Porto</b>	1-0	Braga	 
04/12	Feirense	0-2	<b>Arouca</b>	 
	<b>Moreirense</b>	3-1	Nacional	 
	Estoril Praia	1-1	Belenenses	 
	V. Guimarães	1-1	Chaves	 
05/12	<b>Paços Ferreira</b>	2-1	Boavista	 

Figura 2.6: Informação de uma jornada ocorrida (vista através de um browser)

```
1 02/12 Marítimo 2-1 Benfica FG
2 03/12 Rio Ave 3-1 Tondela FG
3   Sporting 2-0 V. Setúbal FG
4   FC Porto 1-0 Braga FG
5 04/12 Feirense 0-2 Arouca FG
6   Moreirense 3-1 Nacional FG
7   Estoril Praia 1-1 Belenenses FG
8   V. Guimarães 1-1 Chaves FG
9 05/12 Paços Ferreira 2-1 Boavista FG
```

Figura 2.7: Informação de uma jornada ocorrida após aplicação de um Web Crawler em NodeJS

```

2  <resultados>
3    <jogo>
4      <equipa_casa>CSM</equipa_casa>
5      <resultado>2-1</resultado>
6      <equipa_fora>SLB</equipa_fora>
7    </jogo>
8    <jogo>
9      <equipa_casa>RAC</equipa_casa>
10     <resultado>3-1</resultado>
11     <equipa_fora>CDT</equipa_fora>
12   </jogo>
13   <jogo>
14     <equipa_casa>SCP</equipa_casa>
15     <resultado>2-0</resultado>
16     <equipa_fora>VDS</equipa_fora>
17   </jogo>
18   <jogo>
19     <equipa_casa>FCP</equipa_casa>
20     <resultado>1-0</resultado>
21     <equipa_fora>SCB</equipa_fora>
22   </jogo>
23   <jogo>
24     <equipa_casa>CDF</equipa_casa>
25     <resultado>0-2</resultado>
26     <equipa_fora>FCA</equipa_fora>
27   </jogo>
28   <jogo>
29     <equipa_casa>MFC</equipa_casa>
30     <resultado>3-1</resultado>
31     <equipa_fora>CDN</equipa_fora>
32   </jogo>
33   <jogo>
34     <equipa_casa>GDE</equipa_casa>
35     <resultado>1-1</resultado>
36     <equipa_fora>CFB</equipa_fora>
37   </jogo>
38   <jogo>
39     <equipa_casa>VFC</equipa_casa>
40     <resultado>1-1</resultado>
41     <equipa_fora>GDC</equipa_fora>
42   </jogo>
43   <jogo>
44     <equipa_casa>PDF</equipa_casa>
45     <resultado>2-1</resultado>
46     <equipa_fora>BFC</equipa_fora>
47   </jogo>
48 </resultados>

```

Figura 2.8: Informação de uma jornada ocorrida após a aplicação do script em Perl com REgex

<a href="#">&lt; anterior</a>	JORNADA 17				<a href="#">seguinte &gt;</a>
13/01	Arouca	20:30	Estoril Praia	 h2h	
14/01	Benfica	16:00	Boavista	 h2h	
	V. Setúbal	16:00	Nacional	 h2h	
	Chaves	18:15	Sporting	 h2h	
15/01	Feirense	20:30	V. Guimarães	 h2h	
	Marítimo	16:00	Paços Ferreira	 h2h	
	FC Porto	18:00	Moreirense	 h2h	
	Belenenses	20:15	Rio Ave	 h2h	
16/01	Braga	20:00	Tondela	 h2h	

Figura 2.9: Informação de uma jornada futura (vista através de um browser)

1	13/01	Arouca	20:30	Estoril Praia	h2hSportTv 1
2	14/01	Benfica	16:00	Boavista	h2hBTv 1
3		V. Setúbal	16:00	Nacional	h2hSportTv 1
4		Chaves	18:15	Sporting	h2hSportTv 1
5		Feirense	20:30	V. Guimarães	h2hSportTv 1
6	15/01	Marítimo	16:00	Paços Ferreira	h2hSportTv 1
7		FC Porto	18:00	Moreirense	h2hSportTv 1
8		Belenenses	20:15	Rio Ave	h2hSportTv 1
9	16/01	Braga			

Figura 2.10: Informação de uma jornada futura após aplicação de um Web Crawler em NodeJS



```

2  <jogos>
3      <jogo>
4          <equipa_casa>VDS</equipa_casa>
5          <data>09/12/2016</data>
6          <equipa_fora>GDE</equipa_fora>
7          <odd_casa>1.00</odd_casa>
8          <odd_empate>1.00</odd_empate>
9          <odd_fora>1.00</odd_fora>
10     </jogo>
11     <jogo>
12         <equipa_casa>CFB</equipa_casa>
13         <data>10/12/2016</data>
14         <equipa_fora>CSM</equipa_fora>
15         <odd_casa>1.00</odd_casa>
16         <odd_empate>1.00</odd_empate>
17         <odd_fora>1.00</odd_fora>
18     </jogo>
19     <jogo>
20         <equipa_casa>GDC</equipa_casa>
21         <data>10/12/2016</data>
22         <equipa_fora>MFC</equipa_fora>
23         <odd_casa>1.00</odd_casa>
24         <odd_empate>1.00</odd_empate>
25         <odd_fora>1.00</odd_fora>
26     </jogo>
27     <jogo>
28         <equipa_casa>BFC</equipa_casa>
29         <data>10/12/2016</data>
30         <equipa_fora>VFC</equipa_fora>
31         <odd_casa>1.00</odd_casa>
32         <odd_empate>1.00</odd_empate>
33         <odd_fora>1.00</odd_fora>
34     </jogo>
35     <jogo>
36         <equipa_casa>CDN</equipa_casa>
37         <data>11/12/2016</data>
38         <equipa_fora>CDT</equipa_fora>
39         <odd_casa>1.00</odd_casa>
40         <odd_empate>1.00</odd_empate>
41         <odd_fora>1.00</odd_fora>
42 </jogo>

```

Figura 2.11: Informação de uma jornada futura após a aplicação do script em Perl com REgex

De notar uma estrutura preparada para colocação das respetivas **ODD's** dos jogos embora não aplicado (a ser falado em trabalho futuro).

### - *Várias fontes*

A informação relativamente a **jogadores indisponíveis** por lesão ou castigo não foi encontrada estando completa em nenhuma fonte. Pelo que a sua implementação teve que ser algo manual.

```
1  <indisponiveis>
2    <indisponivel>
3      <nome>JONAS</nome>
4      <equipa>SLB</equipa>
5    </indisponivel>
6    <indisponivel>
7      <nome>MAREGA Moussa</nome>
8      <equipa>VSC</equipa>
9    </indisponivel>
10   <indisponivel>
11     <nome>GRIMALDO Alex</nome>
12     <equipa>SLB</equipa>
13   </indisponivel>
```

Figura 2.12: Informação de jogadores indisponíveis

Perante o facto, adotamos um método que ao interpretar uma jornada, altera o *status* de cada jogador para **1 (Disponível)** e depois lê este ficheiro XML alterando o *status* dos jogadores presentes nele para **0 (Indisponível)**. O objetivo seria reformular este ficheiro XML para cada jornada.

## 2.3 Critérios de decisão

Após a descrição da base de conhecimento da secção anterior e em jeito de resumo conseguimos obter estas informações:

- Lista de equipas de uma determinada liga;
- Lista de jogadores de uma determinada equipa;
- Informações sobre os jogadores (nome, idade, posição, disponibilidade (status) e valor);
- Lista de jogos passados e futuros de uma determinada equipa.

Com estas informações foram delineados os seguintes critérios para decisão de resultado de um determinado jogo **em que o valor positivo estaria ligado com a equipa visitada e o valor negativo com a equipa visitante**, valor esse que será a base da resposta final dada:

- **(50%)Diferença de valores de melhores jogadores** entre as duas equipas seguindo o seguinte modelo:  
 $1 \times \text{GR} + 4 \times \text{DEF} + 5 \times \text{MED} + 1 \times \text{AVA} + 1/2 \times \text{JogadorSuplente} + 1/2 \times \text{JogadorSuplente}$  estando eles disponíveis, ou seja, com o status igual a 1.  
Valores expectáveis exemplo [1023 Boavista vs 1109 Benfica] Diferença de **-86**.
- **(20%)Fator Casa (50) vs Fora(0)**
- **(10%)Dias de descanso/Media de idades jogadores**, ou seja, uma formula que relaciona-se os dias de descanso de uma equipa entre jogos com a média de idades dos jogadores.  
Media de idades dos jogadores = M  
 $\text{Resultado} = \text{ROUNDUP}(\text{Dias de Descanso} * 4 - M * (0,25))$
- **(15%)Classificação das equipas na liga:**  
Classificação Equipa Casa = C  
Classificação Equipa Fora = F  
Pontos equipa casa = F  
Pontos equipa fora = C  
Ou seja, jogando a Equipa A (1º classificado) contra a Equipa B (12º classificado), a distribuição ficaria: **+12**
- **(5%)Motivação**
  - Se 1º = 10 pontos;
  - Se tiver a menos de 3 pontos da posição a cima ou a baixo = 10 pontos;
  - Se tiver em 2º/3º e jogar em casa contra o primeiro = 20 pontos

## 2.4 Desenvolvimento da solução

A ideia passaria em atribuir, aos agentes, funções de recolher informação sobre os resultados passados, informações sobre o jogo a prever, qual a equipa que iria jogar em casa como também informações sobre os jogadores de cada equipa. Estas informações podem ser encontradas ou facilmente calculadas através dos ficheiros *XML* referidos.

Os agentes passariam a informação recolhida para um outro agente, que depois de um processamento final apresentaria a informação ao cliente.

### 2.4.1 Arquitetura

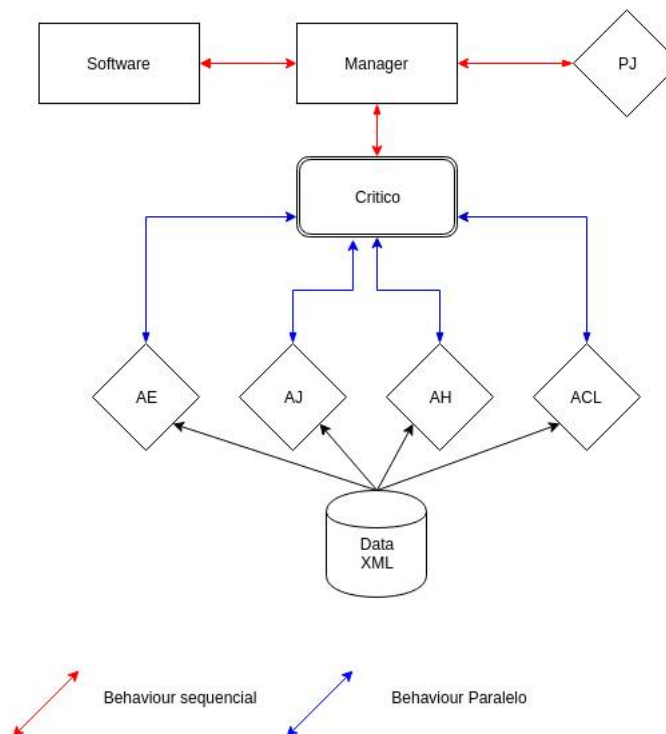


Figura 2.13: :Arquitetura do projeto

Como é possível observar na imagem há vários agentes (*AE*, *AJ*, *AH*, *ACL*) a recolher informações de vários ficheiros *XML* anteriormente referidos.

Depois dos vários agentes terem lido os vários ficheiros e processado a informação lida, sendo que diferentes agentes podem ler o mesmo ficheiro, é enviada essa informação para um agente (*Critico*) que irá reunir as diferentes informações, calcular o peso de cada critério e enviar assim as suas previsões sobre os resultados da jornadas em questão que irá acontecer.

## 2.5 Descrição de Agentes - Modelo Físico

### 2.5.1 Agentes *Man1* e *PJ*

O agente *Man1* (Manager) recebe do agente *Software* o nome da liga que o utilizador escolheu, depois pede ao agente *PJ* a lista dos jogos que estão por acontecer nessa liga, desde a data que o utilizador pediu a análise de resultados. Quando obtiver essa informação irá enviar essa lista ao agente *Critic* e irá esperar até que o agente lhe envie os resultados. Quando o agente *Critic* obtiver essa informação irá enviar uma mensagem ao agente *Man1* que a reencaminhará para o *Software* que será o responsável por apresentar essa informação ao utilizador. O agente *Man1* irá ter um *Behaviour Cyclic* que irá ser o responsável por enviar o nome da liga ao agente *PJ* e consequentemente receber os jogos em falta que o agente *PJ* enviar.

Agente Origem	Agente Destino	Content	ContentObject	Tipo	Ontologia	Obs	
Software	M1	Nome da Liga	-	REQUEST	INFOLIGA	pede informação da liga passando o nome	M1
M1	PJ	-	LigaB	REQUEST	JOGOS	manda uma LigaB vazia	
PJ	M1	-	LigaB	INFORM	JOGOS	recebe uma LigaB com as Predictions com o resultado a 0	
M1	Critic	-	LigaB	REQUEST	CRITIC	manda a LigaB que recebeu do PJ	
Critic	M1	-	LigaB	INFORM	CRITIC	manda uam LigaB com os resultados ja calculados	
M1	Software	-	LigaB	INFORM	INFOLIGA	manda a LigaB ja pronta para mostrar quem ganha	

Figura 2.14: Agente Man1

Agente Origem	Agente Destino	Content	ContentObject	Tipo	Ontologia	Obs	
M1	PJ	-	LigaB	REQUEST	JOGOS		PJ
PJ	M1	-	LigaB	INFORM	JOGOS		

Figura 2.15: Agente PJ

### 2.5.2 Agente *Critic* (Crítico)

O agente *Critic* irá enviar a informação sobre a liga para os outros agentes, em paralelo, para obter informações sobre o histórico das duas equipas, o local onde os jogos irão acontecer (casa/fora), informação sobre os jogadores (média de idade e *skill/valor*) de cada equipa, informação sobre os pontos de cada equipa e posição na tabela classificativa e a data do ultimo jogo de cada equipa.

O agente *Critic* irá ter um *CyclicBehaviour* que é o responsável por receber informações enviadas do agente *Man1* onde invocará um *ParallelBehaviour* para fazer a procura da informação em paralelo.

Agente Origem	Agente Destino	Content	ContentObject	Tipo	Ontologia	Obs	
M1	Critic	-	LigaB	REQUEST	CRITIC		Critic
Critic	AE	EquipaA: EquipaB	-	REQUEST	ESTADO		
Critic	AJ	EquipaA: EquipaB	-	REQUEST	JOGADORES		
Critic	AH	EquipaA: EquipaB	-	REQUEST	HISTORICO		
Critic	ACL	EquipaA: EquipaB	-	REQUEST	CLASS		
AE	Critic	Float	-	INFORM	ESTADO		
AJ	Critic	Float	-	INFORM	JOGADORES		
AH	Critic	Float	-	INFORM	HISTORICO		
ACL	Critic	Float	-	INFORM	CLASS		

Figura 2.16: : Agente *Critic*

### 2.5.3 Agente *AE*

Procura informações sobre a data do último jogo das duas equipas e calcula a média de idade dos jogadores da duas equipas enviando essa informação para o agente *Critic*.

Apresenta um *CyclicBehaviour* para receber o nome das equipas que se irão defrontar e depois invoca um *OneShotBehaviour* com as informações recolhidas.

Agente Origem	Agente Destino	Content	ContentObject	Tipo	Ontologia	Obs	
Critic	AE	EquipaA: EquipaB	-	REQUEST	ESTADO		AE
AE	Critic	Float	-	INFORM	ESTADO		

Figura 2.17: : Agente *AE*

### 2.5.4 Agente *AJ*

Calcula a *skill* dos jogadores de cada equipa (os "melhores" seguindo a tática definida nos critérios 1GR-4DEF-5MED-1AVA mais 3 suplentes), soma a *skill* de todos os jogadores que foram convocados e divide pelo número de jogadores que foram convocados.

Apresenta um *CyclicBehaviour* para receber o nome das equipas que se irão defrontar e depois invoca um *OneShotBehaviour* com as informações recolhidas.

Agente Origem	Agente Destino	Content	ContentObject	Tipo	Ontologia	Obs	
Critic	AJ	EquipaA: EquipaB	-	REQUEST	JOGADORES		
AJ	Critic	Float	-	INFORM	JOGADORES		AJ

Figura 2.18: : Agente *AJ*

### 2.5.5 Agente *AH*

Agente estruturalmente implementado para encontrar na base de conhecimento encontros passados das duas equipas. Calcula o histórico de resultados das duas equipas bem como resultados de jogos entre eles.

Apresenta um *CyclicBehaviour* para receber o nome das equipas que se irão defrontar e depois invoca um *OneShotBehaviour* com as informações recolhidas.

Agente Origem	Agente Destino	Content	ContentObject	Tipo	Ontologia	Obs	
Critic	AH	EquipaA: EquipaB	-	REQUEST	HISTORICO		
AH	Critic	Float	-	INFORM	HISTORICO		AH

Figura 2.19: : Agente *AH*

## 2.5.6 Agente *ACL*

Calcula a posição de cada equipa, ou seja, o local que cada equipa se posiciona na tabela. Apresenta um *CyclicBehaviour* para receber o nome das equipas que se irão defrontar e depois invoca um *OneShotBehaviour* com as informações recolhidas.

Agente Origem	Agente Destino	Content	ContentObject	Tipo	Ontologia	Obs	
Critic	ACL	EquipaA: EquipaB	-	REQUEST	CLASS		
ACL	Critic	Float	-	INFORM	CLASS		ACL

Figura 2.20: : Agente *ACL*

## 2.5.7 Agente *Software*

É o responsável por enviar a liga que o utilizador escolheu e depois apresenta a informação recolhida pelos diversos agentes ao utilizador.

Apresenta dois *CyclicBehaviour* um para enviar qual a liga que o utilizador escolheu e outro que recebe as informações recolhidas pelos agentes.

Agente Origem	Agente Destino	Content	ContentObject	Tipo	Ontologia	Obs	
Software	Man1	Liga	-	Inform	-	Informação qual a liga que o utilizador deseja	
Man1	Software		LigaB	-	-	Resultado da procura	Software

Figura 2.21: : Agente *Software*

## 2.6 Apresentação de Resultados

Após todos os cálculos efetuados mediante os critérios de decisão e o seu peso em termos de percentagem (relembrando que um valor positivo é alusivo à equipa visitada e um valor negativo à equipa visitante) é apresentado o resultado previsto do jogo seguindo os seguintes parâmetros:

- Se resultado estiver entre  $[1, 5[$  ou  $]-5, -1]$  a equipa visitada/visitante, respetivamente, é vencedora;
- Se resultado estiver entre  $[5, +\infty[$  ou  $]-\infty, -5]$ , a equipa visitada/visitante, respetivamente, é super-vencedora (grande probabilidade de acertar);
- Se resultado estiver entre  $[-1, 1]$ , é previsto um empate entre as duas equipas;



### 2.6.1 Funcionamento do programa

Ao carregar no botão *run* do *Netbeans* irá aparecer a seguinte janela.



Figura 2.22: : Menu inicial do programa

Depois de o agente ter calculado toda a informação irá apresentar essa informação ao utilizador.



Figura 2.23: : Resultado da previsão

Através da plataforma *JADE* é possível observar a troca de mensagens entre os agentes. A figura abaixo ilustra um excerto dessa comunicação.

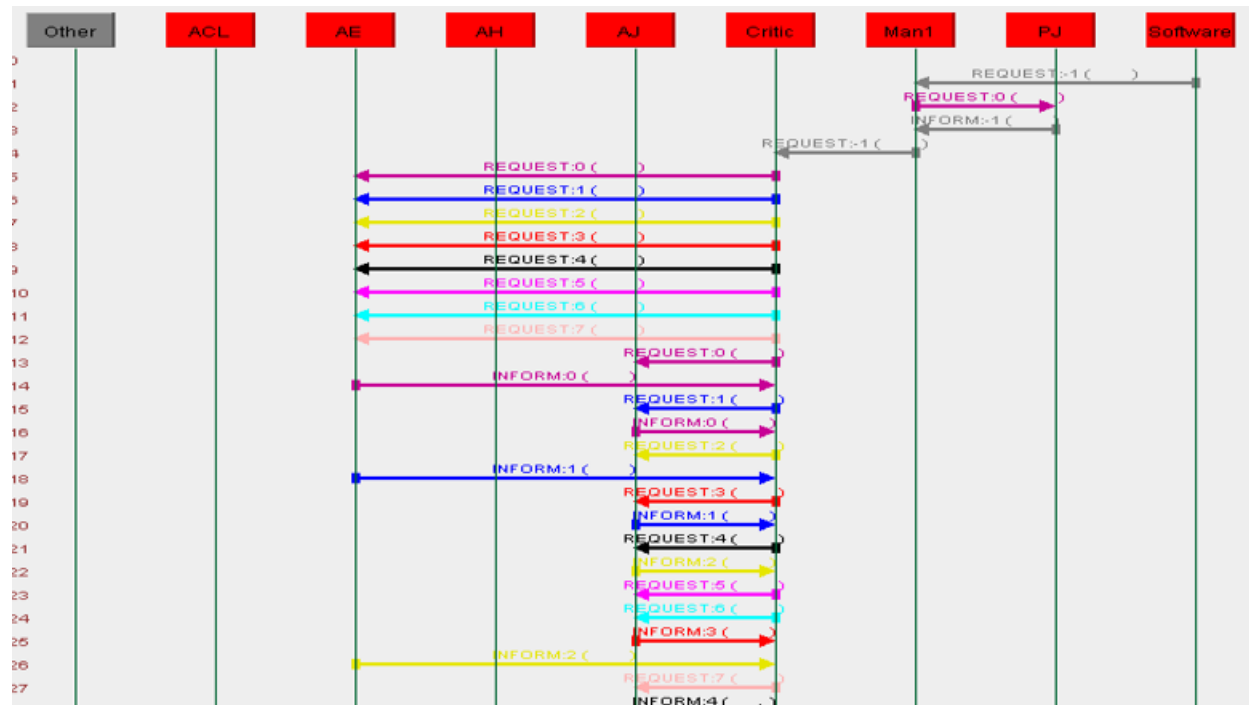


Figura 2.24: : Troca de mensagens entre agentes

## 2.6.2 Preparação do programa

Antes de iniciar o programa primeiramente terá de iniciar a plataforma *JADE*, depois iniciar o *Netbeans* e criar um projeto chamado **AI** com o botão direito do rato aceder as propriedades do projeto e carregar onde diz *run* e mudar a *Main class* para *jade.Boot*. De seguida é necessário mudar os *Arguments* para:

**-container Man1:ai.Man1;PJ:ai.PJ;Critic:ai.Critic;AE:ai.AE;AH:ai.AH;AJ:ai.AJ;ACL:ai.ACL;Software:ai.Software**, sendo que *ai* é o nome da pasta onde se encontram os ficheiros.

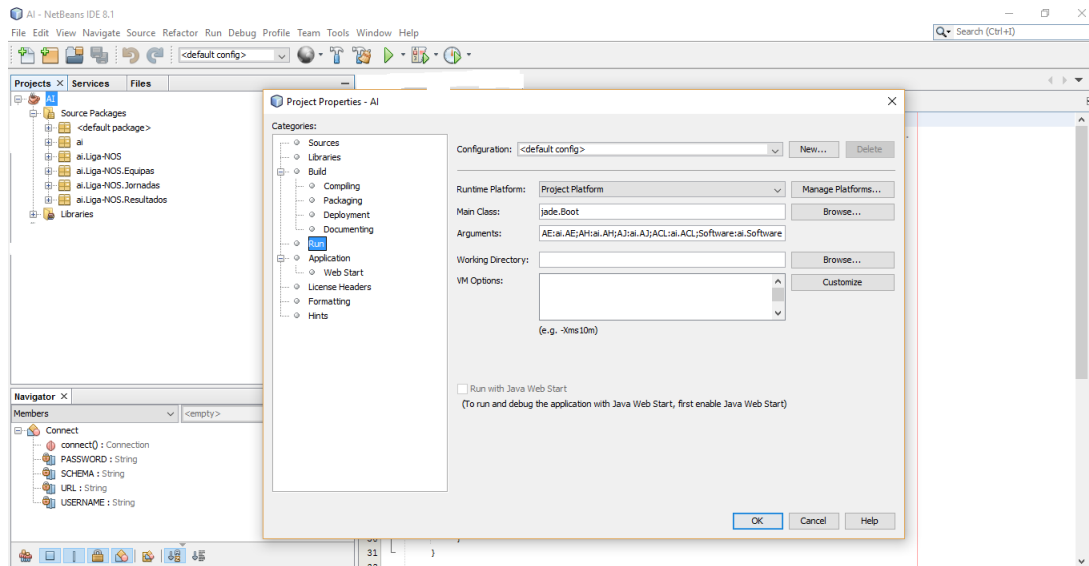


Figura 2.25: : Preparação do programa a sua execução

## 2.6.3 Correr o programa

Depois de ter feito a preparação do programa, para o executar é só carregar no botão *start*, do *Netbeans* e o programa irá abrir um menu inicial com as ligas que dá para escolher, a única liga que dá para escolher é a portuguesa que é a que está presente na base de conhecimento. Ao escolher a liga que pretende obter as previsões irá aparecer um botão e ao carregar nesse botão, passado algum tempo, irá aparecer as previsões dos agentes das diferentes jornadas da liga selecionada.

Nota: Para o botão aparecer é preciso selecionar a liga portuguesa e carregar duas vezes sobre a caixa.

## 2.7 Conclusão e trabalho futuro

Com o aumentar da procura de sistemas capazes de funcionarem de maneira autónoma, ficou cada vez mais claro que a utilização de agentes para auxiliar a tomada de decisão é a escolha mais apropriada, visto que eles podem tomar decisões sozinhos sem que haja interação humana. Um agente pode ser capaz de desempenhar um ou vários papéis necessários ao resolver ou executar uma tarefa, como pesquisa ou até aprendizagem, o que o torna diferente de um simples programa.

Sendo os agentes capazes de desempenhar vários papéis e de trabalhar em conjunto, em sistemas multi-agente, tornasse ainda mais importante o modo como são utilizados e o modo como comunicam entre si, de maneira a tirar proveito de todos os recursos que são possíveis disponibilizar a cada tipo de agente diferente. Devido à quantidade de informação espalhada por diferentes sites a necessidade de ter a informação centralizada acaba por ser benéfico para os diferentes utilizadores que utilizam a plataforma. Por isso de forma a poupar tempo aos utilizadores na procura de qual deveria ser a melhor equipa a apostar decidimos criar a plataforma de previsão de resultados.

Para **trabalho futuro**, ficam os seguintes pontos:

- Refinar base de conhecimento de equipas (e.g Nível do treinador, tática usada, número de adeptos, ...)
- Refinar base de conhecimento de jogadores



Figura 2.26: Perfil Jogador

- Elaborar critérios mais detalhados baseados em novos dados destas bases de conhecimento
- Interligação com ODD's de sites de apostas
- Histórico de previsões com percentagens de certas ou erradas em eventos passados

## 2.8 Bibliografia

<http://www.pucrs.br/facin-prov/wpcontent/uploads/sites/19/2016/03/tr013.pdf>

<http://arquivo.ulbra-to.br/ensino/43020/artigos/anais2002/EIN2002/EIN-2002-Arquiteturas.pdf>

<http://www.devmedia.com.br/introducao-aos-sistemas-multiagentes/28973>

[Novais e Analide,2006] Paulo Novais,Cesar Analide, "Agentes Inteligentes",Universidade do Minho.

[Novais e Analide,2006] Paulo Novais,Cesar Analide, "Agentes Inteligentes",Universidade do Minho.(slides).