Universidade do Minho

Mestrado em Engenharia Informática

2011

# Sistemas Inteligentes

## Sistemas Autónomos

## Programação de Robôs – Competição RoboCode

Ano Lectivo de 2011/2012

pg20189 **André Pimenta Ribeiro**
pg19824 **Cedric Fernandes Pimenta**
pg20978 **Rafael Fernando Pereira Abreu**

5 de Junho de 2012

**Resumo**

O presente relatório tem como objectivo expôr o estudo realizado na construção de processos de simulação de comportamentos individuais e de grupos sociais, no ambiente de programação **RoboCode**.

Serão expostos todos os comportamentos implementados na linguagem *JAVA*, assim como a justificação e enquadramento de cada um, tanto no âmbito individual como no âmbito colectivo.

# Conteúdo

# Lista de Figuras

# Capítulo 1

# Introdução

## 1.1 Contextualização e Apresentação do Problema

Cada vez mais verificamos que a utilização de sistemas autónomos são uma realidade e uma necessidade do mundo actual, pela capacidade que estes têm de perceber o ambiente em que se encontram e, através de sensores, reagir e adaptar-se ao mesmo.

No caso de estudo deste trabalho prático, pretende-se que através do ambiente de programação **RoboCode** e da capacidade de simulação que este oferece através das diferentes classes de robôs de simulação, a implementação e utilização de diferentes estratégias de inteligência artificial para o desenvolvimento de robôs individuais, bem como de equipas de robôs (robôs colectivos), onde se possam estudar e aplicar técnicas de cooperação, de execução de tarefas em grupo e de tomada de decisão.

Serão apresentados, numa fase inicial, robôs que se caracterizam e manifestam através de características que representam, dentro do máximo possível, emoções, sentimentos e comportamentos tipicamente humanos.

Posteriormente, serão apresentados comportamentos sociais, ou seja, comportamentos a apresentar e que se impõem num ambiente onde se necessita de comunicação e colaboração para atingir um objectivo comum, um objectivo de equipa.

Também será abordado, neste mesmo estudo e tanto a nível dos robôs individuais como de equipa, o modelo **OCC** (*Ortony, Clore, Collins, "The Cognitive Structure of Emotions"*) que servirá para representar as emoções dos respectivos robôs, assim como a sua reacção a determinados estímulos e acções provenientes do meio, meio este que será o campo de batalha.

Tudo isto será implementado e representado através da linguagem de programação *JAVA*, linguagem capaz de fornecer todas as ferramentas necessárias para a construção dos já mencionados robôs.

## 1.2 Motivação e Objectivos

Este projecto tem como principal motivação, a aprendizagem e utilização de estratégias de inteligência artificial que permitem o desenvolvimento de sistemas autónomos. Mais concretamente, a implementação de determinadas características em robôs que terão de reagir e interpretar o ambiente à sua volta de forma automatizada e eficaz. Estas características têm como principal foco a representação de comportamentos humanos, assim como a sua manifestação ligada a estímulos e mudanças provocadas por eventos que ocorram no ambiente de interação.

A utilização e criação de tácticas e estratégias de comunicação para os robôs também é um

outro grande objectivo deste trabalho, pois a interseção, cooperação e comunicação trazem uma complexidade muito maior ao problema apesar de possuir a vantagem de trazer uma maior valia que é a socialização e colaboração de diferentes agentes (robôs) para um objectivo comum. E, tal como é natural e caso seja bem definido, estas características trazem inúmeras vantagens pois torna-se possível abandonar a limitação de ser uma única entidade a realizar uma tarefa para o uso de uma equipa que trabalha em direção a um objectivo em comum.
Um outro aspecto motivacional neste trabalho é a utilização do modelo **OCC** (*Ortony, Clore, Collins, "The Cognitive Structure of Emotions"*) no ambiente **RoboCode** e os respectivos resultados e aplicações reais que poderemos conseguir na conjunção dos mesmos.

## 1.3 Estrutura

O presente relatório é constituído por 4 capítulos. Sendo este o primeiro, denominado **Introdução**, onde se contextualiza e apresenta o problema, bem como a nossa motivação e objectivos para o trabalho. No capítulo seguinte, de nome **Preliminares**, apresentaremos os conhecimentos que consideramos necessários para que o leitor possa obter uma melhor compreensão deste documento enquanto no capítulo III (**Desenvolvimento**) é explicado tudo o que foi desenvolvido durante o nosso trabalho, os resultados obtidos e as decisões tomadas. Por fim, no capítulo final (**Conclusão**) são exprimidas as nossas opiniões sobre o projecto e, tendo como base os resultados finais, é efectuado o balanço do trabalho.

# Capítulo 2

# Preliminares

## 2.1 Domínio do Problema

Neste capitulo serão apresentados os diversos conceitos usados no problema, assim como o domínio do problema e a sua organização.

## 2.2 Robocode

Tal como já foi referido, iremos trabalhar no ambiente de programação **RoboCode**, dotado de diferentes classes que permitem a implementação de robôs distintos que apresentam diversas características e capacidades. Apresentam-se, de seguida ,as diferentes classes de robos.

- **Robot** – classe base para robôs;
- **Droid** – interface que implementa robôs sem radar mas com mais energia;
- **AdvancedRobot** - classe avançada de robôs que permite um maior controlo e liberdade de implementação;
- **TeamRobot** - classe que permite a implementação de robôs de equipa, assim como a sua comunicação.

## 2.3 Arquiteturas de controlo

Os diferentes robôs regem-se por diferentes arquiteturas de controlo que definem, de certa forma, o seu comportamento ao longo do seu ciclo de vida.
Para uma melhor percepção destas arquiteturas, apresenta-se uma pequena descrição do seu funcionamento:

- ***Arquitetura Deliberativa*** - Esta arquitetura baseia-se na deteção dos dados sensoriais resultantes do ambiente, que irão ser usados para planear as acções a efectuar de seguida.
- ***Arquitetura Reativa*** - Esta arquitetura tem como principal destaque a rapidez de resposta pois omite o planeamento, limitando-se a um conjunto de regras condição-acção pré-programadas, também sendo mínima a representação do ambiente.

- ***Arquitetura Híbrida*** - Esta arquitetura é uma mistura das duas anteriores, aplicando sistemas reactivos no controlo de baixo nível e sistemas deliberativos ao nível da tomada de decisão.

- ***Arquitetura Comportamental***. Esta arquitetura inspira-se na biologia, e procura imitar comportamentos animais na resolução de problemas. Baseia-se num conjunto de comportamentos cuja a finalidade é a de alcançar ou manter determinados objectivos mas que são influenciados por determinadas emoções desencadeadas por eventos do ambiente.

## 2.4 Estratégias de controlo

O sistema de controlo, no caso em questão aplicado aos robôs no ambiente **RoboCode**, é responsável pela utilização da informação recolhida do ambiente e calcular a forma como os robôs vão interagir com o ambiente.
Apresentam-se de seguida as estratégias de controlo utilizadas:

- ***Open Loop***. Esta estratégia baseia-se na não utilização de sensores, sendo a sua utilização benéfica em ambientes estáticos e/ou previsíveis.

- ***Feedforward***. Esta estratégia baseia-se na utilização de sensores para a recolha de informação. Esta informação será utilizada para actualizar as variáveis que modelam o ambiente.
  Apresenta uma fraca utilidade quando o ambiente é dinâmico.

- ***Feedbackward***. Esta estratégia utiliza os sensores para monitorizar o ambiente, continuamente, ajustando o robô conforme a interpretação do mesmo ambiente. Este estratégia permite uma constante actualização do mundo externo.

## 2.5 Modelo OCC

O modelo **OCC** (*Ortony, Clore, Collins, "The Cognitive Structure of Emotions"*) é utilizado para a representação de diferentes emoções que são definidas no modelo e que se vão adaptando e transformando em resposta a eventos tanto positivos como negativos, acções e objectos.
O modelo é descrito e apresentado, de forma detalhada, em [1].

# Capítulo 3

# Descrição do Trabalho e Análise de Resultados

## 3.1 Robos individuais

Nesta secção serão apresentados os robôs individuais, desenvolvidos com o objectivo de representar um conjunto de características, comportamentos e emoções que os robôs podem vir a demonstrar ao longo de uma batalha.

As emoções que os robôs vão expondo ao longo do seu ciclo de vida (batalha) são demonstradas usando o modelo **OCC**.

## 3.2 Modelo OCC no RoboCode

Uma das nossas principais decisões tomadas foi a aplicação do modelo **OCC** introduzido durante este semestre. O modelo aplicado não foi exatamente o modelo **OCC** mas sim uma derivação deste, que introduz conceitos um pouco diferentes do original mas que fazem muito sentido, este fator foi suficiente para nos convencer dos benefícios que ele traria para o nosso projeto.

Explicando detalhadamente como este modelo foi aplicado durante o nosso desenvolvimento, temos que inicialmente é feita a caracterização do robô segundo as qualidades/defeitos que pretendíamos simular que este tivesse (a sua personalidade). Assim, isto é decidido através da atribuição de valores a 5 variáveis:

```
private double openness = 0.87;
private double conscientiousness = 0.63;
private double extraversion = 0.22;
private double agreeableness = 0.48;
private double neuroticism = 0.42;
```

Estas características são, então, traduzidas para um universo de 3 variáveis, designado por **PAD**, *Pleasure* e *Arousal* e *Dominance*, e que tomam valores entre -1 e 1 pela aplicação de fórmulas que usam os 5 valores acima mencionados. Através deste universo, foi possível distinguirmos doze emoções (seguindo como referência o trabalho efetuado pelos autores do artigo mencionado na bibliografia). Estas doze emoções foram escolhidas tomando os seguintes valores:

```
public static Point3D joy = new Point3D(0.4, 0.2, 0.1);
public static Point3D hope = new Point3D(0.2, 0.2, -0.1);
```

```
public static Point3D relief = new Point3D(0.2, -0.3, 0.4);
public static Point3D pride = new Point3D(0.4, 0.3, 0.3);
public static Point3D gratitude = new Point3D(0.4, 0.2, -0.3);
public static Point3D love = new Point3D(0.3, 0.1, 0.2);
public static Point3D distress = new Point3D(-0.4, -0.2, -0.5);
public static Point3D fear = new Point3D(-0.64, 0.6, -0.43);
public static Point3D disappointment = new Point3D(-0.3, 0.1, -0.4);
public static Point3D remorse = new Point3D(-0.3, 0.1, 0.6);
public static Point3D anger = new Point3D(-0.51, 0.59, 0.25);
public static Point3D hate = new Point3D(-0.6, 0.6, 0.3);
```

Por fim, é importante referir que de modo a que uma nova emoção atue no nosso robô, segundo este modelo, a intensidade dessa emoção necessita de ser maior do que o módulo da diferença entre *extraversion* e *neuroticism*. Para além disto, após uma emoção influenciar um robô, com o decorrer do tempo, este tende para voltar ao seu estado inicial, segundo a fórmula:

$$I(t) = I * e^{(-t*n)}$$

onde n é o *neuroticism*, t é o tempo que decorreu desde que a emoção foi despoletada e I é a intensidade da nova emoção. Com estes fatores todos a atuarem sobre o robô, ficamos bastante satisfeitos com o resultado e somos da opinião que conseguimos expôr com precisão as diferentes emoções que são despoletadas no robô quando o ambiente provoca determinados eventos (por exemplo, quando o robô dispara um projétil, a nova emoção será esperança de acertar no inimigo; se falhar, a nova emoção será desapontamento mas se acertar será alegria). Através disto, podemos ter uma melhor simulação das ações do sistema quando um determinado conjunto de fatores atuam sobre este (a título de exemplo se, a cada vez que for atingido, a raiva aumentar, ao atingir determinados níveis, o robô pode abandonar a sua estratégia e deixar-se levar pela emoção).

### 3.2.1 Odómetro

Estes robôs possuem também um odómetro que regista a distancia que os robôs percorrem ao longo do campo de batalha, permitindo assim analisar de uma certa forma o desempenho dos robôs, ou melhor, permite analisar a necessidade de deslocação que estes procuram em relação aos resultados que obtêm.

```
public void move(double vel){
                setAhead(vel);
                odometro+=vel;
}
```

Contagem do movimento efectaudo pelos robôs.

### 3.2.2 Perseguição

A definição deste robô foi inspirada na definição da palavra *Perseguição*. Retiraram-se algumas características dessa definição e elas forma implementas no robô, tais como: *insistência, persuasão, cuidado.*

| | |
|---|---|
| Estratégia de controlo | **Feedbackward** |
| Arquitetura de controlo | **Deliberativa** |
| Funcionalidades | • Persegue<br>• Se for mais fraco ataca (racismo)<br>• Se for mais forte segue de longe<br>• Alvo fixo |

### 3.2.3 Fuga

Este robô, por sua vez, foi inspirado na definição da palavra *Fuga*, onde se deu destaque às seguintes características: *atenção, rapidez, astúcia.*

| | |
|---|---|
| Estratégia de controlo | **Feedbackward** |
| Arquitetura de controlo | **Deliberativa** |
| Funcionalidades | • Fugir sempre dos outros robôs<br>• Sempre em alerta<br>• Analisar o seu redor<br>• Fugir aos 'S's<br>• Se não valer a pena fugir, ataca para recuperar energia |

### 3.2.4 Pontaria

Da definição de pontaria, as seguintes características foram retiradas para a implementação do respetivo robô: *precisão, astuto, concentração* .

| | |
|---|---|
| Estratégia de controlo | **Feedbackward** |
| Arquitetura de controlo | **Deliberativa** |
| Funcionalidades | • Adquire o comportamento de um "sniper"<br>• No início da batalha dirige-se para um canto e aí dispara segundo um algoritmo preditivo (linear ou circular) dependendo do número de balas falhadas<br>• No fim da batalha, verifica se quando morreu, ainda existiam muitos robots vivos e, em caso afirmativo, muda de canto<br>• As emoção necessitam de ter uma grande intensidade para atuarem sobre os robots pois, neste cenário, estes têm personalidades fortes |

### 3.2.5 Valentia

Da definição de valentia retiraram-se as seguintes características a implementar no robô: *energia, força, vigor, resistência.*

| | |
|---|---|
| Estratégia de controlo | **Feedfoward** |
| Arquitetura de controlo | **Reactiva** |
| Funcionalidades | • Atacar todos<br>• Não ter medo<br>• Não medir consequências<br>• Reage a quem se mete com ele |

### 3.2.6 Medo

A definição de medo é uma das mais subjetivas e complexas, razão pela qual a escolha de características não foi propriamente fácil, resultando nas seguintes características: *atenção, sensibilidade, reação*

| | |
|---|---|
| Estratégia de controlo | **Feedbackward** |
| Arquitetura de controlo | **Reactiva** |
| Funcionalidades | • Sempre alerta<br>• Sem iniciativa<br>• Foge quando se metem com ele<br>• Desvia-se de robôs mais fortes ao deslocar-se<br>• Procura zonas vazias |

### 3.2.7 Timidez

A timidez vai ser representada em forma de *falta de coragem, insegurança, acanhamento, inibição.*

| | |
|---|---|
| Estratégia de controlo | **Feedbackward** |
| Arquitetura de controlo | **Reactiva** |
| Funcionalidades | • Evitando a colisão com as paredes<br>• Fica corado (vermelho) quando colide com o adversário<br>• Quando deteta outro robô, tenta fazer operações de evasão (fugir) |

### 3.2.8 Susto

O susto é algo que deriva um pouco do medo e de alguma timidez, pelo que se dicidiu caracterizar esta característica pelos estados de *sobressalto, terror repentino, receio.*

| Estratégia de controlo | **Feedfoward** |
|---|---|
| Arquitetura de controlo | **Reactiva** |
| Funcionalidades | • Ajusta a potência do disparo consoante a distância do adversário mais perto<br>• Quando os seus valores do modelo PAD estão perto de (-0.50,30,-0.45), fica muito assustado:<br>– Fica pálido (Branco)<br>– Roda há sua volta e dispara com a máxima potência para todos os sítios |

### 3.2.9 Audácia

O robô audaz irá ser caracterizado pelo *impulso de alma que leva a cometer ações extraordinárias, desprezando obstáculos e perigos.*

| Estratégia de controlo | **Feedbackward** |
|---|---|
| Arquitetura de controlo | **Híbrida** |
| Funcionalidades | • Dispara sempre com a potência máxima e persegue o robot com mais vida<br>• Quando perde muita vida (fica com $< 50$), adota um comportamento mais cauteloso:<br>– Tenta se desviar das balas e dispara segundo um algoritmo preditivo |

### 3.2.10 Temor

Da definição de temor podemos destacar algumas características que se conjugam com a definição de medo, no entanto decidiu-se destacar *Ato ou efeito de temer, viver no temor da miséria, da velhice, da morte.*

| | |
|---|---|
| Estratégia de controlo | **Feedbackward** |
| Arquitetura de controlo | **Reactiva** |
| Funcionalidades | • Teme pela sua vida e foge, sempre que pode, da possibilidade de morrer:<br>– Teme os outros robôs (que podem causar a sua morte) (30 <= energia < 80): caminha para a parede mais proxima<br>– Quando se aproxima de um estado mais decadente (energia < 30): caminha para o canto mais próximo |

### 3.2.11 Atrevimento

Da definição de atrevimento, este robô será caracterizado por uma *insolência e ousadia.*

| | |
|---|---|
| Estratégia de controlo | **Feedbackward** |
| Arquitetura de controlo | **Deliberativa** |
| Funcionalidades | • Procura o robot com mais vida e persegue-o disparando sobre este com a potência máxima enquanto não deteta outro robot com mais vida<br>• Adota sempre o mesmo comportamento |

### 3.2.12 Petulância

Caracterizou-se petulância como a *vivacidade impetuosa e ousadia* e decidiu-se implementar estas características no respetivo robô.

| | |
|---|---|
| Estratégia de controlo | **Feedbackward** |
| Arquitetura de controlo | **Deliberativa** |
| Funcionalidades | • Não foge quando as balas lhe acertam<br>• Dispara sobre o robot com mais vida no campo de batalha usando um algoritmo preditivo<br>• Possui sempre o mesmo comportamento |

### 3.2.13 Apreensão

Este é o robô que tem como objectivo representar características de apreensão e, para isso, atribui-se características como *receio, cisma, preocupação, apreensão diante do desconhecido.*

| | |
|---|---|
| Estratégia de controlo | **Feedbackward** |
| Arquitetura de controlo | **Deliberativa** |
| Funcionalidades | • Enquanto consegue, vai tentar andar em frente evitando as paredes de forma a conhecer todo o campo de batalha<br>• Dispara sempre que avista ou bate contra outro robô, efetuando uma fuga em seguida<br>• Tem medo do que está para lá do conhecido (paredes), por isso evita-as |

## 3.3 Exemplos de batalhas de alguns robôs apresentados

No exemplo a seguir podemos observar o robô pontaria que se deslocou para um canto e tenta eliminar os seus alvos:

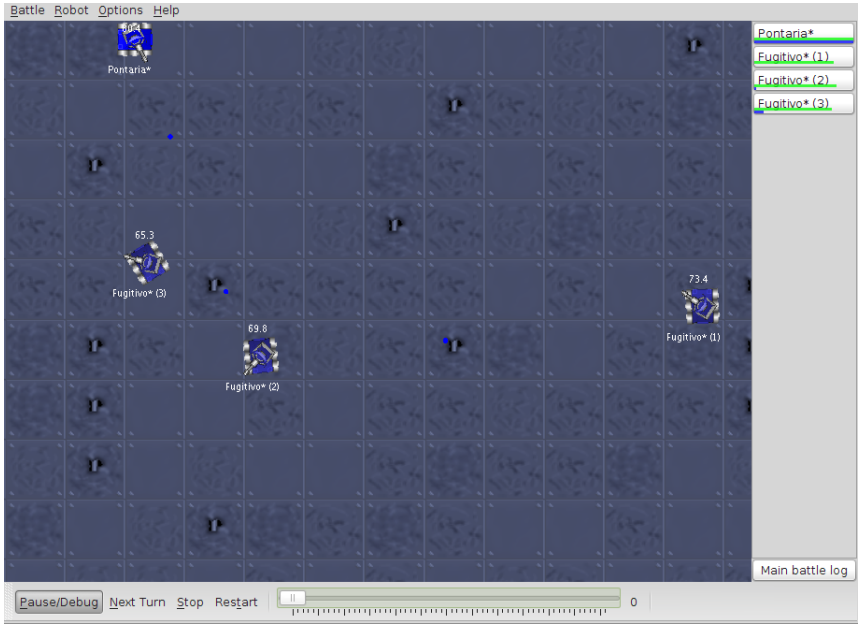


**Figura 3.1:** Robô Pontaria

Neste exemplo, o robô susto tenta desmarcar-se para uma zona onde se encontrem menos robôs de forma a não ficar no meio do fogo da batalha:

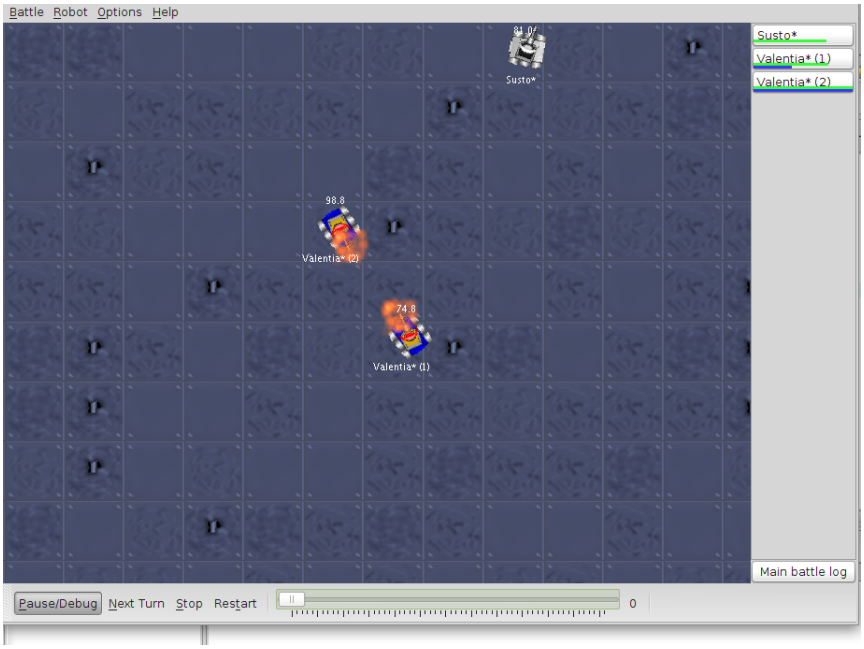


**Figura 3.2:** Robô Susto

No exemplo seguinte podemos ver um conjunto de **robôs perseguição** a seguir, à distancia, o **robô medo**, tentado não o perder de vista.


**Figura 3.3:** Robô Perseguição

## 3.4 Equipas de robôs

Nesta secção serão apresentados os robôs desenvolvidos para trabalhar em equipa, assim como as suas tácticas e estratégias, tanto de comunicação como de colaboração, para atingir um fim, sendo neste caso específico a vitória no campo de batalha.
Será dado um enfoque na forma de comunicação dos robôs bem como na hierarquia de tarefas que são atribuídas a cada um e o seu papel no campo de batalha.

### 3.4.1 Comunicação entre robôs

A comunicação entre os elementos de uma equipa é uma das peças fulcrais para o êxito da mesma, assim como a forma de comunicação utilizada.
Através de uma comunicação clara e eficiente é possível efetuar uma distribuição de tarefas tendo em vista um objectivo comum e, sobretudo, ter uma maior fonte de informação sobre o mundo externo; este fator permite que a comunicação entre os elementos de uma equipa orientada à partilha de conhecimento seja efetuada de forma muito mais completa, sendo que esta informação é conseguida apartir de diferentes locais simultâneamente. No entanto, a comunicação de equipa possui um lado perigoso e que facilmente se pode tornar prejudicial caso existam falhas de comunicação, comunicação de informação errada ou informação incompleta.
Para que se realize uma comunicação, do modo mais efeciente, seguro e objectivo possível, foi tido em conta o domínio do problema e foi definida uma *ontologia* baseada no ambiente de programação **RoboCode** em junção com conceitos de base militar.

#### Ontologia

Para realizar uma comunicação eficiente entre os diferentes elementos de uma equipa, decidiu-se criar e definir uma *ontologia* que se baseia em conceitos militares, adaptando estes ao ambiente em que desenvolvemos o nosso projeto.
A ontologia é, então, definida pelas seguintes classes:

- **Attack**. Este *comando* tem como objectivo ordenar um ataque a determinada posição do campo de batalha ou a determinado inimigo.
  É definido por uma prioridade de operação bem como um alvo de ataque.
- **Help**. Este *comando* tem como objectivo ordenar a defesa de determinado elemento da equipa.
  É também definido por uma prioridade que tem como objectivo definir a necessidade de suporte relativamente a outros comandos. Contem informação relativa ao robô que está a causar dano e à localização onde a ajuda é necessária.
- **Move**. Este *comando* define um movimento de determinada posição para outra e tem como principal objectivo definir movimentos estratégicos a serem executados por qualquer elemento da equipa.
- **Target**. *Classe* que define um alvo (possui o nome do alvo, a posição, a sua energia e o seu tipo).
  Esta informação é usada para enviar alvos de ataque para os diversos elementos da equipa.

- **Position**. *Classe* que define um posição 2D no campo de batalha do ambiente **RoboCode**.
- **Shot**. *Classe* que precisa a informação para o disparo de um projétil.
  Tem como objectivo a utilização para disparos letais.
- **Command**. *Classe* com comandos simples, mais concretamente, para comunicação por *Strings*. (ex: informação por frases)

### 3.4.2 Robôs de equipa

Um dos objectivos deste trabalho é a definição e criação de equipas de robôs que colaborem para o alcance de um objectivo comum. No nosso ambiente de trabalho (**Robocode**), este objectivo em comum traduz-se na vitória numa batalha contra outras equipas de robôs.

Foi decidida a implementação de uma estratégia militar, baseada numa hierarquia militar. Esta hierarquia é baseada em experiência militar e sobretudo nas capacidades e conhecimentos dos seus elementos, o que permite assumir que um superior deve ser mais capaz e dotado de mais conhecimento e estratégia do que o elemento abaixo dele na hierarquia.
Existem diversos cargos militares, assim como hierarquias militares, sendo que optámos por nos basear na hierarquia militar portuguesa, escolhendo alguns cargos de forma a poder representar esta na equipa de robôs.

Apresentam-se de seguida os cargos adotados para a equipa.

O **General** da equipa será o *Team Leader* e será dotado das maiores responsabilidades no campo de batalha, estando constantemente a dar instruções aos membros da equipa, assim como a analisar o campo de batalha e decidir que táctica a equipa deve adotar.

O **Sargento** será um elemento da equipa que assume algumas responsabilidades e que é dotado de alguma liberdade tacticamente sendo capaz de tomar algumas decisões tanto a nivel individual como colectivo.

O **Atirador** será o elemento que terá as características mais específicas de toda a equipa. Pode ser considerado um elemento especial, que tem como objectivo "atirar para matar".

O **Soldado** será o elemento que mais se arriscará no campo de batalha e dará sempre a cara neste mesmo campo de batalha. Tem como objectivo estar na linha da frente aquando da batalha.

O **Cabo** será o elemento que se limita a obedecer ao **General** e que tem como objectivo realizar as tarefas que este ordena, priorizando isto em detrimento de tudo o resto.

#### Estratégias de Combate

Tendo em conta que o objectivo do ambiente **RoboCode** passa por eliminar os inimigos e, por consequente, ganhar a batalha, o uso de tácticas, especialmente em combates de equipa, torna-se fundamental para que se possa ter vantagem sobre os diferentes adversários.

Optou-se por **não** construir diferentes equipas para diferentes estratégias, mas sim preparar a equipa para se adaptar com diferentes estratégias ao longo da batalha, tendo sempre em

conta os diferentes cenários possíveis de batalha e os eventos que ocorrem neste ambiente.

Num cenário de superioridade numérica, a equipa adopta uma estratégia de ataque, tentando aproveitar a superioridade numérica nesse momento e fazendo uma recuperação de terreno, forçando o inimigo a recuar.

Já perante um cenário de inferioridade tanto numérica, como de condição/energia da equipa, o líder adota uma estratégia de deslocação para uma zona de batalha que tenha menos inimigos, chamando assim uma estratégia de **movimento**. Esta estratégia, no caso de inferioridade, acaba por anteceder uma estratégia defensiva, regularmente.

Quando se opta pela estratégia defensiva, a equipa retira-se para uma posição onde se procura uma formação mais compacta e que dê protecção ao "Team Leader" e ao atirador, devido às funções especiais que estes assumem e que foram mencionadas acima.

### Tomada de Decisões

A tomada de decisões, tal como já referido, é feita maioritariamente pelo líder da equipa. O **General** toma estas decisões tendo sempre em conta o factor do número de elementos da equipa e dos adversários bem como o nível de energia das equipas.

No caso de morte do **General** a liderança passa para o seu inferior directo, neste caso o **Sargento**, sendo que este é mais limitado no processo de decisão e de comando. A partir de Sargento e para baixo na hierarquia, deixa de existir qualquer tipo de liderança, passando a equipa a ficar desorientada e cada um por si (tomada de decisões a nível individual).

Na figura abaixo pode-se ver a equipa a dar proteção ao lider, ou seja, a reagir a um comando de ajuda dado pelo mesmo. (**General**).


**Figura 3.4:** Cenário de batalha

#### Sistema de Prioridades

A existência de múltiplos eventos e tarefas que ocorrem ao longo da batalha no **RoboCode** pode tornar a tomada de decisões muito difícl, chegando mesmo a perturbar as acções dos robôs nessa batalha.

De forma a tentar optimizar, do melhor modo, o funcionamento destas e pesando, também, o sistema de hierarquia, implementou-se um sistemas de prioridades. Este tem como intuito destacar o tipo de ordens e a hierarquia mas sem esquecer das simples tarefas que necessitam de ser efetuadas em prol da equipa e que desempenham um papel muito importante.

As prioridades serão definidas numa escala de 1 a 5, cabendo a cada robô interpretar e escalonar, da melhor forma possível, estas mesmas prioridades.

De seguida segue-se um exemplo de um controlo de prioridades, neste caso relativo à energia de um robô.

```
public int priority(double energy){
        if (energy > 20) {
                return 5;
        } else if (energy > 40) {
                return 4;
        } else if (energy > 60) {
                return 3;
        } else if (energy > 80) {
                return 2;
        } else if (energy > 100) {
                return 1;
        }
return 0;
}
```

### 3.4.3 Classes de Robôs

Apresentaremos agora as diferentes classes implementadas que traduzem os robôs já descritos assim como a apresentação de algumas características que os distinguem entre si.

#### Robô Soldado

Ao contrário do **Robô Cabo**, o **Robô Soldado** já é dotado de um radar que o permite ter uma percepção melhor do ambiente em que se insere, ou neste caso do campo de batalha.

O **Robô Soldado**, tal como já foi apresentado, é um robô de primeira linha e que tem como missão entrar na primeira linha de batalha e atacar o inimigo. Possui ainda muitas das características ideais para este tipo de militar, como a audácia e a coragem.

No entanto, e tal como o **Robô Cabo**, em caso de emergência este passa a ter como prioridade a protecção dos seus superiores. Obedece a comandos de movimento e ataque caso tenham uma prioridade elevada.

### Robô Cabo

Este é um robô que implementa a classe *Droid*, que não é possuidora de radar, ou seja tem uma percepção do mundo limitada mas que em contra partida é compensado com mais energia que qualquer outro robô e garantindo, também, uma energia extra ao seu líder.

O **Robo Cabo**, devido à sua limitação de percepção e sensorização, vai estar ligado directamente ao seu líder sendo guiado cegamente por este.

Como objectivos, terá de atacar as coordenadas que lhe forem passadas, bem como assumir como prioridade máxima as ordens que implicam segurança do líder de equipa.

Sem liderança, este robô fica totalmente desorientado e passa para um estado de pânico.

### Robô Atirador

Este robô é talvez o que se especializa mais dentro da equipa. A sua missão principal é eliminar alvos específicos, focando-se neles até os conseguir eliminar.

A sua missão inicial é eliminar o líder da equipa inimiga, tentando desta forma que fiquem limitados a nível de segurança e que surja, consequentemente, uma desorganização na equipa adversária (desorientação).

Este robô recebe, do seu líder, alvos específicos para abater e, tal como os robôs já apresentados, é possuidor de um sistema de prioridades responsável por decidir se deve mudar de alvo ou se deve terminar a sua missão.

### Robô Sargento

Este é o segundo robô na hierarquia de comando escolhida e que já dispõe de alguma liberdade de escolha e decisão. É capaz de deliberar ordens/comandos caso ache que estes sejam necessários e ainda não tenham sido atribuídos pelo líder pois a sua posição pode lhe dar outra visibilidade do campo de batalha.

Este robô procura eliminar inimigos perto da formação e que se encontrem com um nível de energia baixo, de forma a tentar eliminar da forma mais rápida possíveis danos que estes possam causar na equipa.

### Robô General

Este robô é o *Team Leader*, e como tal, é um robô que devido à extrema importância que tem para a equipa (devido às suas capacidades), assume o papel mais preponderante, podendo ser mesmo considerado o cérebro da equipa .

Este robô efetua uma análise constante e rigorosa de todas as variáveis do campo de batalha e decide, após a análise das mesmas, tanto as tácticas que a equipa deve assumir como as movimentações que deve efectuar.

Devido à sua importância, caso a sua presença no campo de batalha esteja em risco, ele emite um pedido de ajuda aos restantes membros da equipa com uma prioridade alta, de forma a que estes tentem eliminar a ameaça, mesmo que isso cause a sua eliminação (sacrifício por um bem maior, o benefício da equipa).

# Capítulo 4

# Conclusão e Trabalho futuro

O desenvolvimento dos diferentes robôs mencionados ao longo deste relatório, tanto os de nível individual como os de nível colectivo, permitem efetuar análises concretas sobre a utilização de sistemas autónomos bem como as necessidades de que estes padecem.

Desde já, pode-se afirmar que a perceção do mundo exterior, tal como a sensorização do mesmo, é fundamental para o funcionamento de qualquer sistema minimamente autónomo pois, de modo a que seja possível efectuar as suas tarefas de forma correcta, precisa-se de um conhecimento constante do ambiente onde se envolve, salvo a excepção de se tratar de um ambiente previsível ou estático (o que não se verifica no ambiente **RoboCode** que foi utilizado como caso de estudo deste trabalho).

É, também, um facto que para diferentes tarefas, e no caso em questão para diferentes características, se adequam também diferentes estratégias de controlo pois nem sempre é necessário, ou mesmo adequado, o processamento constante de informação.

A nível das equipas de robôs destacou-se a importância de comunicação que estas necessitam para que se obtenha informação com um grau de confiança elevado, e que leve a uma execução correta das tarefas que resultam no sucesso dos objectivos que lhes são definidos. No entanto, esta é uma tarefa complexa e que necessita de muitos cuidados pois quanto mais diversificada for a equipa, e quanto maior for o número de factores que esta leva em consideração, maior é o cuidado exigido para garantir que a informação é sempre correta e para que todos possam ter a informação sempre actualizada de modo rápido e seguro.

Quanto à utilização do modelo **OCC**, é possível destacar-se o modo impressionante como consegue simular as emoções humanas com grande precisão, e no caso do **RoboCode**, torna possível a aproximação entre as componentes deste e os estímulos que estes recebem através do ambiente exterior e o modo como o comportamento pode ser modelado consoante a informação que se possui.

Como trabalho futuro fica a evolução da equipa de robôs, que apesar de tudo se encontrar num nível funcional aceitável, esta sempre sujeita a possíveis melhorias. Importante mencionar o trabalho efetuado no desenvolvimento das premissas de comunicação, apesar de algumas necessitarem de sofrer algumas melhorias.

Com isto, considerámos este projeto como uma boa solução na simulação de sistemas autónomos e, após visualizar os nossos resultados obtidos, sentimo-nos satisfeitos tanto com a colaboração existente nas equipas de robôs como nas ações efetuadas pelos robôs individuais para a obtençãos dos seus objetivos particulares; a aplicação do referido modelo **OCC** contribuiu de forma imensurável para a representação das reações aos eventos do ambiente.

# Bibliografia


[1] Santos R, Marreiros G, Ramos C, Neves J, Bulas-Cruz J.: *Personality, Emotion, and Mood in Agent-Based Group Decision Making* .
IEEE Computer Society, 2011.

[2] Maria Goreti.: *Agentes de Apoio à Argumentação e Decisão em Grupo* .
Universidade do Minho, 2007.

[3] José Maia Neves.: *A LOGIC INTERPRETER TO HANDLE TIME AND NEGATION IN LOGIC DATA BASES.*
Universidade do Minho, 1984.

[4] Susana Rocha Rodrigues.: *Eventos Adversos e Não Conformidades nos Cuidados de Saúde.*
Universidade do Minho, 2009.

[5] Rodrigues S., Brandão P., Nelas N., Neves J., Alves V..: *A logic programming approach to medical errors in imaging.*
International journal of medicalinformatics, 2009.