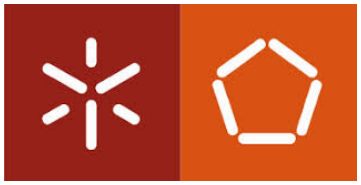

Universidade do Minho
Mestrado em Engenharia Informática

Agentes e Sistemas Multi-Agente

Agentes Cooperativos e Competitivos

Grupo 8

Carolina Santejo	Filipa Pereira	Luís Pinto	Raquel Costa
pg47102	pg46978	pg47428	pg47600

Braga
10 de Maio de 2022

Conteúdo

1 Introdução

O presente relatório surge no âmbito do trabalho prático da unidade curricular Agentes e Sistemas Multi-Agente, pertencente ao perfil de Sistemas Inteligentes do Mestrado em Engenharia Informática, da Universidade do Minho.

1.1 Caso de Estudo

O projeto tem por base a conceção e implementação de um sistema multiagente, desenvolvido num ambiente ***JADE*** (Java Agent Development Framework) [1], com agentes ("inteligentes") capazes de sensorizar e interpretar a informação adquirida do ambiente virtual em que estes habitam e tomar decisões inteligentes, de forma cooperativa ou competitiva, com base num objetivo concreto.

Os intervenientes são os tanques (agentes jogadores), que devem estabelecer processos de negociação e colaboração entre agentes da mesma equipa. Recorrendo, deste modo, a estratégias de procura e posicionamento colaborativas, de forma a melhorar a performance geral de cada equipa.

1.2 Principais Objetivos

No que diz respeito aos principais objetivos, estes consistem em, através dos diferentes modos de ação de cada jogador, permitir a monitorização do terreno de combate pelos tanques. Mais concretamente, explorar o mundo (modo explorador), ter em conta a presença de agentes oponentes (modo defensivo) e perseguir e eliminar cada oponente (modo atacante).

Para a concretização dos objetivos propostos foi necessário definir uma arquitetura capaz de suportar o processo de planeamento e tomada de decisão, estratégias de comunicação e sincronização de agentes, métricas de avaliação de performance, e, por fim, definir funções objetivo de modo a analisar e motivar, respetivamente, o movimento dos agentes.

1.3 Estrutura do Relatório

Este projeto encontra-se dividido em duas fases sendo que na primeira apresenta-se a parte de modelação do projeto, utilizando-se para tal linguagem ***UML*** (Unified Modeling Laguage) adaptada ao novo paradigma de computação baseada em Agentes. Já a segunda fase corresponde à exploração do processo de implementação.

Na secção **2.1** deste documento escrito, é introduzido o **Domínio da Solução**, no qual, através da análise do modelo domínio, se explora as várias entidades que constituem a solução proposta pelo grupo. Por outro lado, ao longo da secção **2.2** aborda-se a arquitetura desenvolvida para o sistema, apresentando-se o diagrama de *Packages* e o diagrama de classes. Em adição, na secção **2.3** é detalhada, com o auxílio da máquina de estados e de diagramas de atividades e colaboração, a interação existente entre os agentes que constituem o sistema. Na secção **3** será explicada a implementação

destes mesmos agentes salientando-se a função objetivo na secção **3.1**, as ferramentas de visualização na secção **3.2** e efetuando-se, também, a avaliação do sistema e testes de performance ao mesmo (secção **3.3**).

Por fim, este relatório termina com uma secção de conclusões (secção **4**) onde é feita uma visão crítica do trabalho realizado, e com uma proposta de trabalhos futuros (secção **5**).

2 Design e Modelação em AgentUML

2.1 Domínio da Solução

Com a realização deste trabalho o grupo pretendeu construir um sistema multi-agente, composto por duas equipa de cinco agentes cada. Cada agente é tanto colaborativo como competitivo, sendo capaz de comunicar com os elementos da sua equipa (colaboração) de modo a cumprir o seu objetivo, eliminar agentes da equipa adversária (competição). O tabuleiro do jogo pode ser definido como um campo de posições com dimensão 35x35. Sendo que, para a deteção, e posterior eliminação, dos agentes ao seu redor, cada agente possui um campo de visão com dimensão 7x7. O processo de eliminação de um agente consiste em rodear o mesmo, em todas as direções de movimento possíveis (cima, baixo, esquerda e direita), por agentes da equipa adversária.

No seguinte diagrama encontram-se esquematizadas as ideias previamente apresentadas.

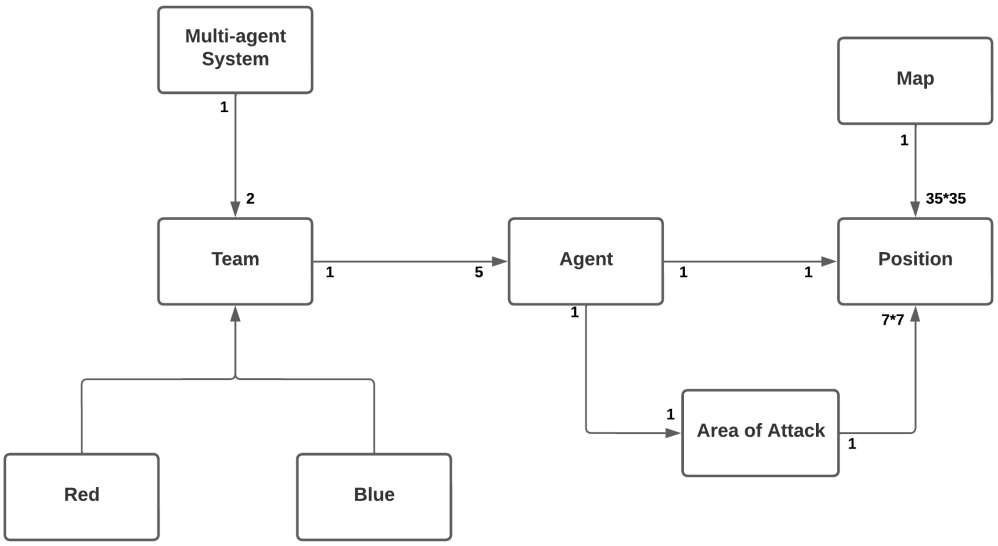

Figura 1: Modelo de Domínio

2.2 Arquitetura do Sistema

Nos próximos tópicos será abordada a arquitetura que o grupo desenvolveu para o sistema, bem como as suas características. Primeiramente, será discutido as várias abordagens possíveis para a construção da arquitetura, sendo possível adotar uma solução descentralizada ou centralizada. De seguida, será apresentada aquela que foi escolhida pelo grupo, tal como os motivos da decisão tomada. Por fim, com o auxílio de diagramas UML é delineada a estrutura da solução.

2.2.1 Arquitetura Adotada

O grupo considerou duas abordagens diferentes para o processo de planeamento da cooperação entre os agentes. Existindo então a arquitetura centralizada e descentralizada. No caso da **centralizada**, existe um único agente que constrói o plano, este é interpretado como sendo um líder e o principal responsável pela tomada de decisão e cooperação entre os agentes. Já em relação à arquitetura **descentralizada** e **distribuída**, esta pressupõe que o plano é construído por mais de um agente, portanto, ao contrário da arquitetura anterior, não existe um único agente que possui uma visão global das atividades do grupo de agentes.

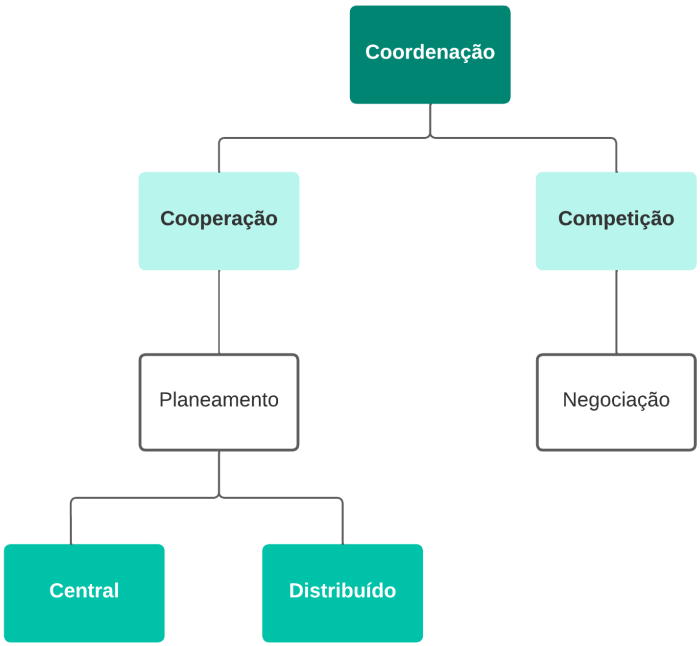

Figura 2: Taxonomia da coordenação

De modo a escolher a arquitetura mais vantajosa para o caso de estudo em questão, o grupo decidiu levantar vantagens e desvantagens de cada uma das abordagens. Sendo que o principal objetivo foi que no final fosse possível realizar um balanço final de forma a auxiliar a escolha sobre a arquitetura mais adequada e que traria mais benefícios ao trabalho. Assim sendo, o grupo chegou à seguinte conclusão:

Tabela 1: **Arquitetura Centralizada**

Vantagens	**Desvantagens**
• Visão do se passa no sistema e dos agentes integrantes • Facilidade na sincronização de mensagens • Simplicidade na implementação • A tomada de decisão é feita apenas por um agente	• Se o agente "líder"falhar então o sistema fica comprometido • Os agentes não têm necessariamente consciência da existência dos outros • Não há distribuição da carga nem das responsabilidades pelos vários agentes • Os agentes apenas "seguem"o agente líder, perdendo autonomia

Tabela 2: **Arquitetura Descentralizada**

Vantagens	**Desvantagens**
• Carga e perícia distribuídas • Os agentes partilham a arquitetura e o software • Os agentes ostentam comportamentos cooperativos • Múltiplos agentes trabalham em prol de um objetivo ao mesmo tempo	• Custos de manutenção podem ser elevados • É necessário ter cuidado com a sincronização de mensagens entre os vários agentes • Estabelecimento de parâmetros de negociação específicos, de forma a todos os agentes chegarem a um acordo mútuo

Posto isto, o grupo decidiu optar por uma arquitetura **descentralizada**, **adaptativa**, e **distribuída**. A adaptabilidade permite que os agentes se adequem conforme o ambiente, sendo que no caso em estudo os agentes podem comportar-se de forma diferente perante diversas situações, podendo assumir, por exemplo, uma posição de ataque ou defesa. Já em relação à distribuição, o grupo considera que é vantajoso a longo termo, as responsabilidades serem divididas pelos vários agentes, de forma a não sobrecarregar um único agente e, assim, fabricar um sistema que não esteja tão propício a falhas. Além disso, estas características permitem realçar a autonomia e colaboração entre os agentes, uma vez que eles são capazes de comunicar diretamente entre si e de tomar decisões em conjunto em prol da equipa. Enquanto que na arquitetura centralizada seria feita uma comunicação assistida onde os vários agentes comunicavam unicamente com o agente líder, e este, sozinho, tomava uma decisão por toda a equipa. Apesar desta abordagem ser mais simples, o grupo considerou que os agentes perdiam um bocado a sua autonomia uma vez que não eram capazes de tomar qualquer tipo de decisão já que seria o líder a tomar por eles. Além disso, o agente líder estaria sobrecarregado com as mensagens dos vários agentes podendo acontecer eventuais atrasos na tomada de decisão.

2.2.2 Estrutura da Solução

Tendo estabelecida a arquitetura do sistema foi necessário definir a estrutura da solução. Desta forma, para representar a organização dos elementos de uma forma mais intuitiva foram elaborados diagramas de *packages* e classes.

Analisando em primeiro lugar o diagrama de *packages*, pode-se observar na figura 3 a existência de 4 conjuntos de classes. No package *Agents* estão localizadas todas as classes correspondentes aos agentes do sistema, sendo que os behaviours do agente *Player* foram agrupados no package *Behaviours*. Por outro lado, no package *Classes* localizam-se todas as classes auxiliares que foram utilizadas sendo que as classes referentes a exceções se colocaram no package *Exceptions*.

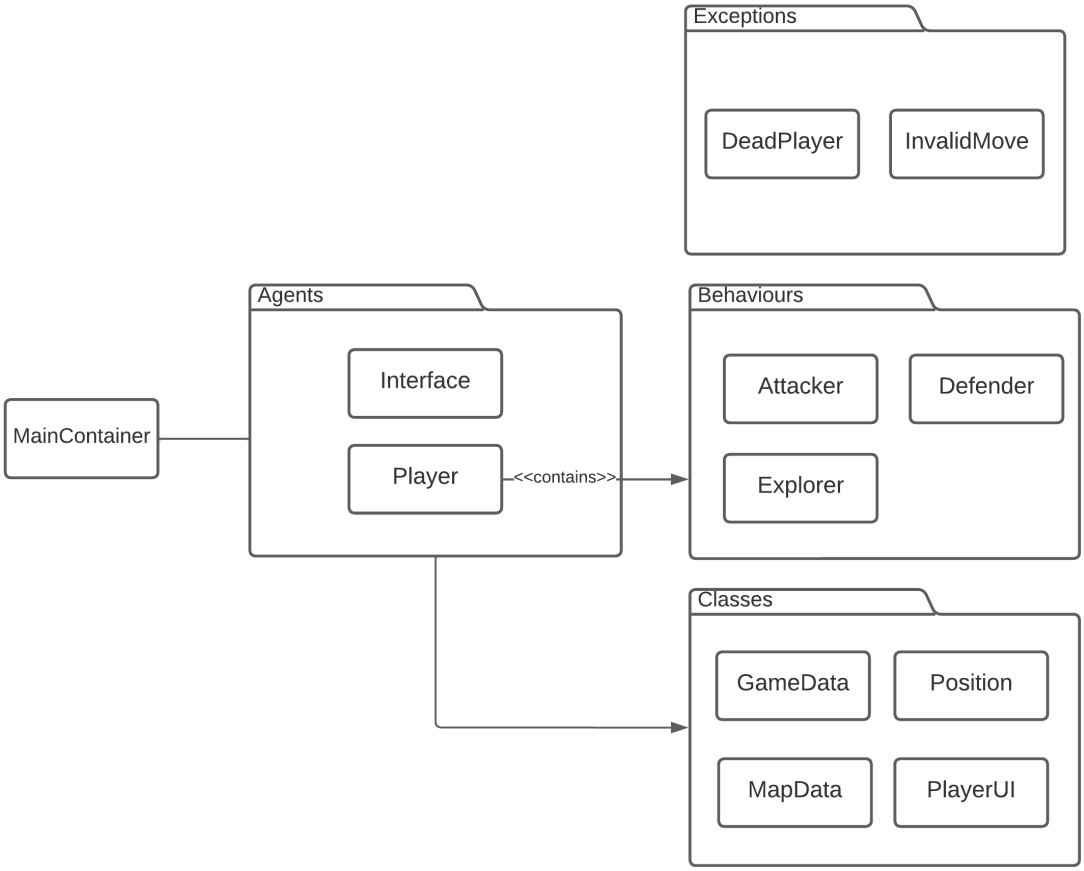

Figura 3: Diagrama de Packages

Passando à análise do diagrama de classes [Fig. 4], é possível identificar dois tipos de Agentes no sistema: *Player* e *Interface*. A classe *Player* representa cada jogador do jogo, sendo que contém todas as informações e métodos relevantes para o mesmo se poder movimentar no mapa. Quanto à classe Interface, esta é responsável por receber todas as alterações dos jogadores e movimentar os mesmos pelo mapa. Para além disso, esta classe implementa todo o código correspondente à interface gráfica do jogo e todas as suas mudanças ao longo do tempo. É de notar que todas as alterações, tanto do

mapa como da interface serão guardadas na classe 'GameData' que é uma variável de instância da 'Interface'.

É também importante realçar que ambos os agentes comunicam entre si através de diferentes *Behaviours*. Desta forma, para conseguir receber mensagens externas, ambos implementam um *Cyclic Behaviour* (classes *UpdateView* e *ReceiveMessages*) que será responsável por intercetar todas as informações recebidas e fazer determinadas ações de acordo com o conteúdo das mesmas. Estas ações podem englobar, não só modificações na classe como também o envio de uma resposta de volta ao remetente.

Por outro lado, para enviar mensagens foram utilizados dois tipos de Behaviour: *Ticker* e *One Shot*. No agente Interface implementou-se o KillPlayer que é acionado quando um jogador é morto. Já no Player existem 2 TickerBehaviour que é importante destacar: *SendPosition* e *Movement*. O primeiro tem como objetivo enviar continuamente o seu campo de visão e posição para todos os jogadores da sua equipa, para que desta forma cada *player* tenha sempre acesso às informações dos *teammates* em tempo real. Relativamente à classe *Movement*, esta tem como objetivo escolher continuamente uma determinada estratégia de jogo de acordo com as informações da equipa no momento. Analisando as diferentes posições e campos de visão será invocado um dos 'OneShot-Behaviour's entre o Explorer, Atacker ou Defender que se adeque melhor à situação do jogo.

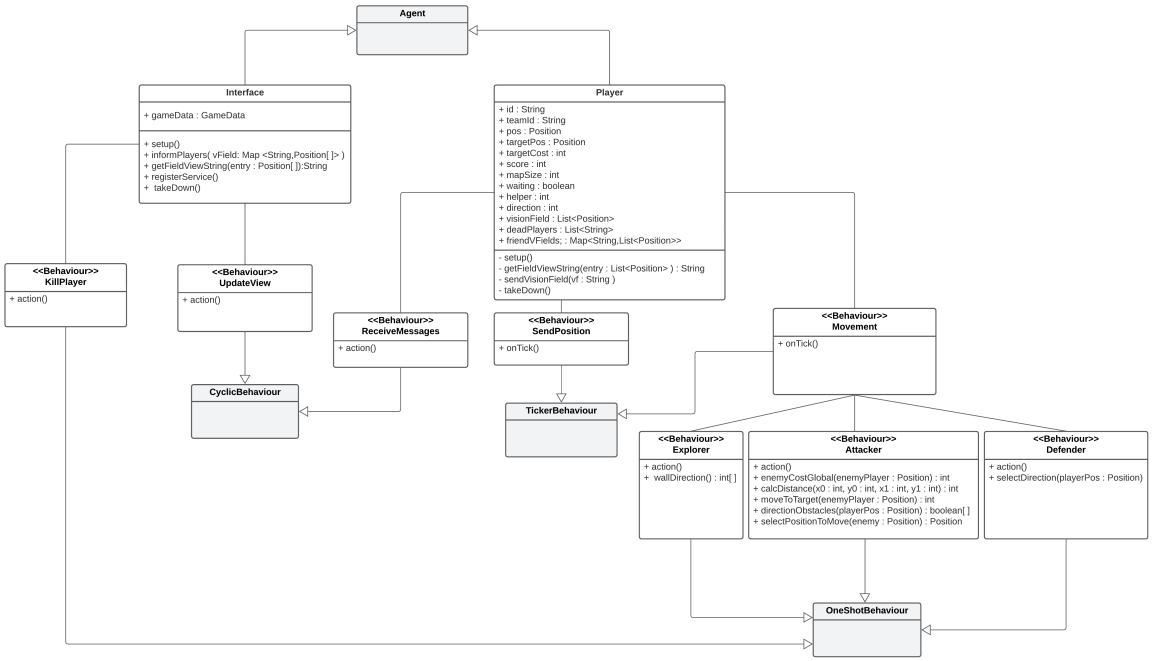

Figura 4: Diagrama de classes

2.3 Interação Entre Agentes

Nesta secção iremos abordar o funcionamento interno e externo de um agente, por outras palavras, pretendemos fornecer uma visão global de como cada agente opera internamente e como o mesmo interage com os restantes comunicadores/intervenientes. Além disto, ao longo desta secção serão também explicadas as três atividades mais relevantes do nosso sistema, sendo elas: explorar, atacar e defender.

2.3.1 Diagrama Colaboração

Em primeiro lugar é necessário entender de um modo mais abstrato o fluxo de trocas de mensagens entre os agentes jogadores de uma equipa. O processo funciona da seguinte maneira: um agente Player depois de calcular a melhor direção para a qual se deve mover, envia esta mesma direção à interface, que depois irá calcular a nova posição do jogador e representá-la no mapa. Além disso, a interface irá verificar se algum agente jogador morreu com esta nova jogada realizada, caso isto se verifique é então enviada uma mensagem ao jogador com essa mesma informação. Caso contrário, é enviada então a nova posição e o novo campo de visão ao jogador para que este possa atualizar esses mesmos campos, e, por fim, enviar em broadcast, para todos os agentes da sua equipa, uma mensagem que contém a sua posição e o seu campo de visão. Deste modo, os jogadores de uma mesma equipa vão estar informados sobre a posição dos seus colegas e sobre os obstáculos e inimigos que estes mesmos encontram.

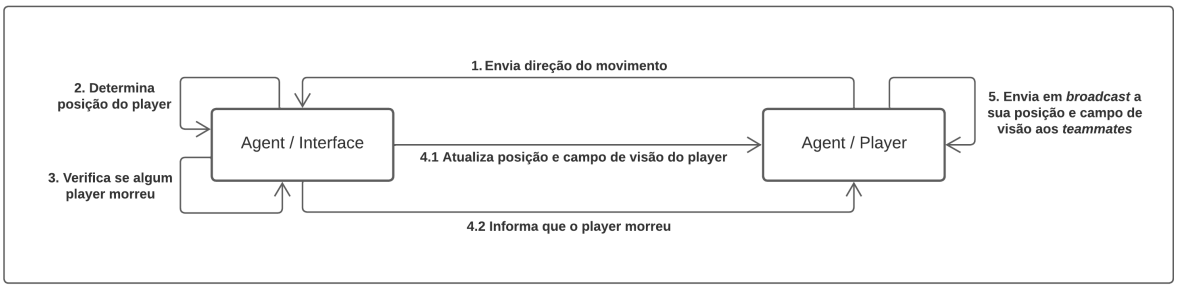

Figura 5: Diagrama de Colaboração

2.3.2 Estados do Agente

Um dos aspetos mais relevantes a salientar é o facto de que os agentes ***Player*** não são estáticos, sendo que poderão alterar o seu estado (estratégia de jogo) ao longo de uma partida. Desta forma, e de acordo com a figura 6, irá ser abordado nesta secção os diversos estados que um jogador poderá adotar.

Primeiramente, assim que o jogo inicia, o jogador adota automaticamente a estratégia de ***Explorer***, apenas movimentando-se pelo mapa. O *player* mantêm-se com este estado caso seu campo de visão continue vazio. Por outro lado, caso um jogador detete no seu campo de visão um inimigo **ou** caso seja chamado por um colega de equipa para auxiliar num ataque, então o jogador em questão adota a estratégia de ***Attacker***. No entanto, se o *player* verifica que no seu campo de visão existe mais do que um inimigo **e** que o número de inimigos é superior ao número de colegas de equipa, então o jogador adota a estratégia de ***Defender***.

Em suma, um jogador vai analisando o seu campo de visão e poderá manter a estratégia atual, ou então mudá-la caso se verifiquem as condições necessárias para o efeito. Todo este processo cessa quando o jogo termina.

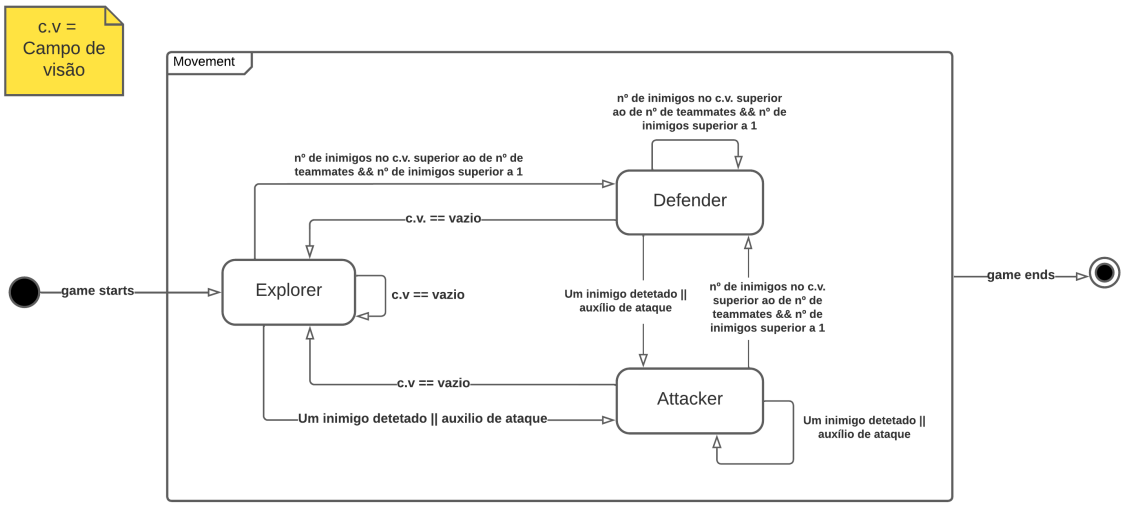

Figura 6: Máquina de estados

2.3.3 Explorar

Neste sistema, o modo de exploração consiste em que o jogador se movimente pelo mapa enquanto que o seu campo de visão se encontre vazio. Tal como podemos observar pela figura 7, a atividade *explorar* inicia com o agente Interface a calcular o campo de visão e a enviá-lo, através de uma mensagem, para o *player* correspondente. Este jogador, analisa o campo de visão que recebeu e caso detete a presença de uma parede, irá querer movimentar-se numa direção no sentido contrário. Esta decisão foi tomada de forma a tentar maximizar o campo de visão, visto que este fica condicionado quando o jogador se desloca junto a uma parede.

Por outro lado, se não forem detetadas bordas, o jogador irá querer movimenta-se para o centro do mapa, de modo a cobrir uma maior área do tabuleiro e possivelmente detetar um inimigo. Posto isto, e para cada um dos casos referidos, o jogador irá escolher uma direção, das várias que poderá ter disponíveis, enviando para a Interface uma mensagem com o pedido de atualização para a nova posição.

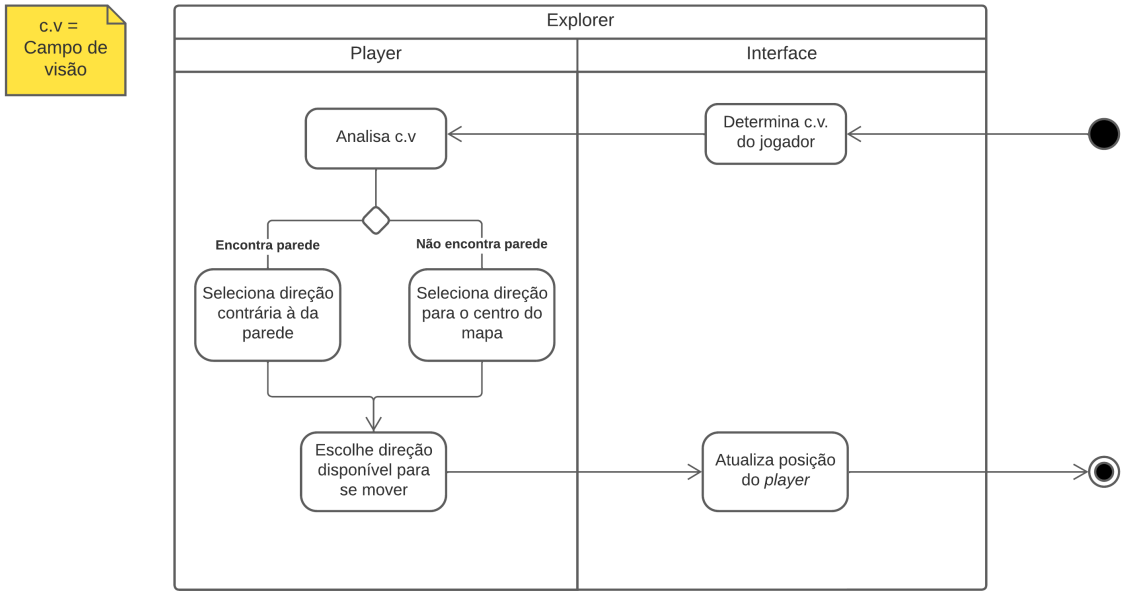

Figura 7: Diagrama de atividades do *Explorer*

2.3.4 Atacar

Tal como podemos observar pela figura 8, a atividade *Ataque* poderá ter dois inícios. O fluxo principal é aquele em que o agente Interface calcula o campo de visão e envia-o, através de uma mensagem, para o *player* correspondente. Este jogador, analisa o campo de visão que recebeu e, se detetar um inimigo no mesmo, calcula o somatório das distâncias dos vários elementos da equipa ao *target*. Se o ataque ao inimigo identificado compensar, ou seja, se o somatório das distâncias for inferior comparativamente ao seu *target* anterior, então será enviado uma mensagem *broadcast* a todos os outros colegas de equipa a pedir auxílio no ataque. Se o novo *target* não compensar, então o estado mantém-se e a atividade é interrompida. Por fim, e tal como acontecia no primeiro fluxo referido, o jogador envia um pedido ao agente Interface para atualizar a sua posição.

Já no segundo fluxo, o jogador recebe uma mensagem de um colega de equipa a pedir auxílio para um ataque, selecionando a direção que minimiza a distância ao novo *target*. Posto isto, envia um pedido ao agente Interface para atualizar a sua posição.

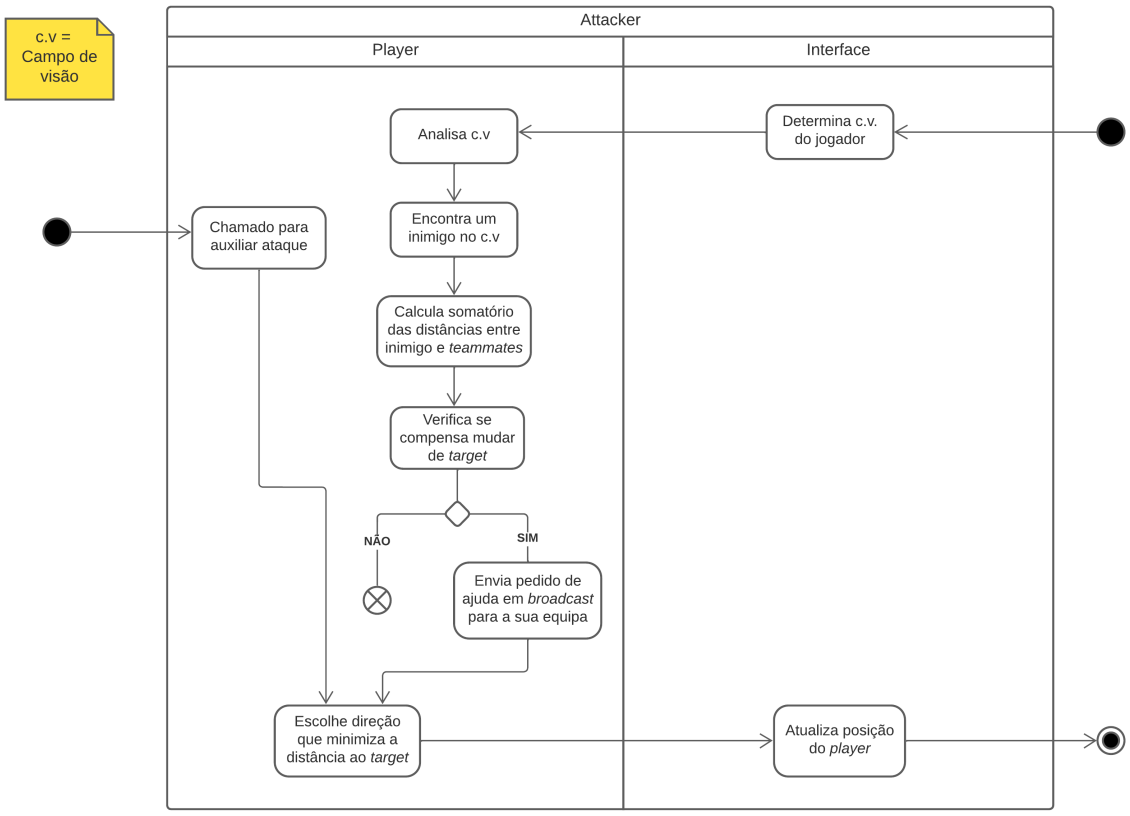

Figura 8: Diagrama de atividades do *Attacker*

2.3.5 Defender

Tal como podemos observar pela figura 9, a atividade *Defender* inicia com o agente Interface a calcular o campo de visão e a enviá-lo, através de uma mensagem, para o *player* correspondente. Este jogador, caso encontre mais do que um inimigo no seu campo de visão, ou seja, caso esteja em desvantagem, irá determinar como se pode afastar do perigo. Para tal, verifica a existência de obstáculos nas posições adjacentes e para as quais se poderá mover. Posto isto, e dentro das direções livres, irá optar por aquela que apresentar um menor número de inimigos. Após isto, é enviado para o agente Interface uma mensagem com o pedido de atualização para a nova posição.

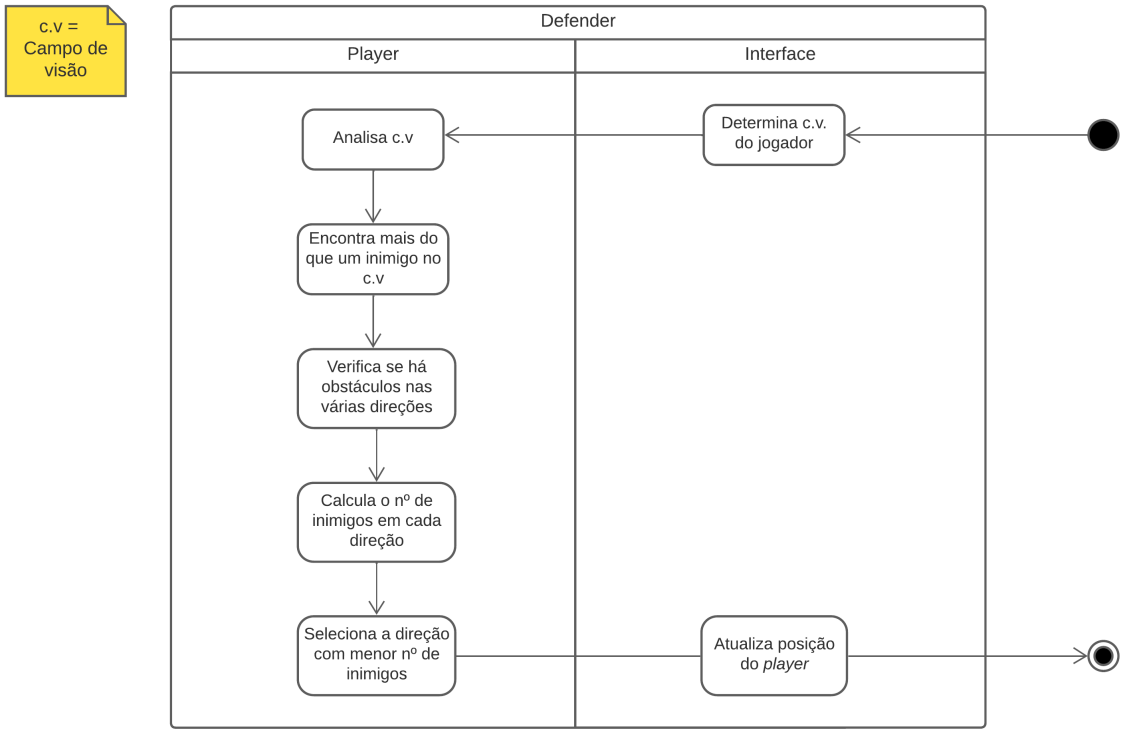

Figura 9: Diagrama de atividades do *Defender*

3 Implementação dos Agentes

3.1 Definição de Funções Objetivo

De modo a possibilitar a negociação entre os agentes e a tomada de uma decisão em equipa consoante as várias situações que podem existir em jogo, foi necessário definir várias funções objetivo, que permitem delinear elementos de decisão e traçar tanto metas como alvos para os agentes em jogo.

Em primeiro lugar, no caso em que o agente adopta um comportamento de "explorador", este tenta afastar-se o máximo possível das paredes, direcionado-se assim para o centro do tabuleiro, onde poderá encontrar inimigos no seu campo de visão e rapidamente alterar o seu comportamento para o de "atacante", ou, no caso de se encontrar numa posição de desvantagem, entrará em modo "defensivo".

Assim que o jogador encontra um inimigo no seu campo de visão, este começa a coordenar o **ataque em equipa** caso o target em questão seja mais vantajoso do que aqueles encontrados por outros membros da equipa. Para tal, é utilizada uma função de custo que calcula o somatório das distâncias entre os vários jogadores da equipa e o inimigo encontrado. Tendo por exemplo a figura [10], é possível reparar que os tanques azuis encontram dois inimigos, portanto é necessário ambos negociarem sobre qual tanque é que devem ambos atacar em conjunto. Um dos tanques azuis tem um tanque inimigo ($x1$) apenas a 3 u.m de distância de si, mas em relação ao outro tanque azul ($y1$) já se encontra a 7 u.m. Já este segundo tanque azul tem um tanque vermelho ($y2$) a 6 u.m de distância de si, e por acaso o seu colega encontra-se a essa mesma distância ($x2$).

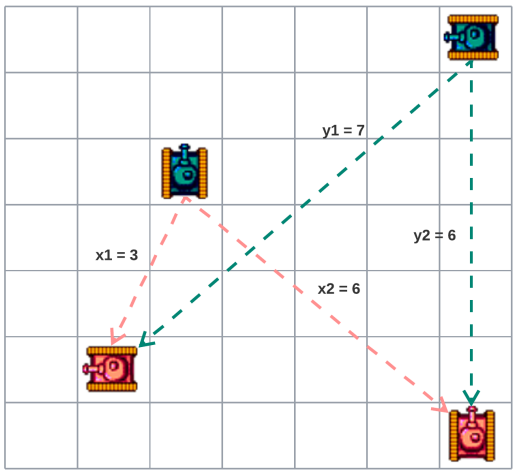

Figura 10: Coordenação do ataque

Aplicando então a função de custo que o grupo definiu obtemos o seguinte:

- Inimigo vermelho 1: $x1 + y1 = 10$
- Inimigo vermelho 2: $x2 + y2 = 12$

Assim sendo, conseguimos concluir que o inimigo 1 é aquele que se encontra mais "perto"de toda a equipa, apesar de ser aquele que se encontra mais longe do segundo tanque azul (y). Desta forma é **reforçado e recompensado o trabalho** em prol da equipa, em vez de os agentes apenas atuarem em beneficio de si próprios.

No caso em que os jogadores entram em **modo defensivo** foi necessário também traçar um objetivo para este tipo de comportamento, de forma a que os agentes conseguissem escapar de situações menos benéficas para si e, consequentemente, para a sua equipa, visto que **perder um elemento é algo que penaliza bastante a própria equipa**.

Portanto, em primeiro lugar, assim que um agente avista no seu campo de visão mais do que um inimigo e o número de colegas de equipa é inferior ao número de inimigos, este entra em modo de defesa. A estratégia delineada foi que o jogador deveria movimentar-se na direção que contém menos inimigos e menos obstáculos tais como as paredes do mapa.

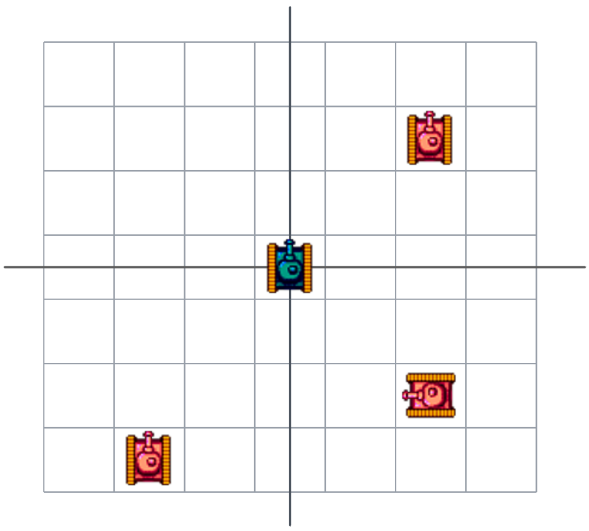

Figura 11: Coordenação da defesa

Considerando como exemplo a figura [11], reparamos que existem 3 inimigos vermelhos a circundar o tanque azul. Aplicando então a função que o grupo definiu, este irá começar por somar o número de inimigos que se encontram à sua direita, isto inclui o 1^{o} e 4^{o} quadrante, ou seja no total tem **2 inimigos à sua direita**. Somando os inimigos que **se encontram abaixo** dele (3^{o} e 4^{o} quadrante), que contém também dois. Tanto **acima de si** como **à sua esquerda** apenas tem um inimigo. Caso os tanques se movimentassem na diagonal, a decisão mais estratégica seria movimentar-se na direção do 2^{o} quadrante, mas como isso não acontece ele deverá movimentar-se ou para cima ou para a esquerda, tendo por base qual o inimigo que se encontra mais distante dele.

A partir destas funções de objetivo apresentadas anteriormente, foi possível definir a negociação que os membros de equipa têm de fazer entre si de modo a que alcançado um acordo mútuo, beneficiando assim a própria equipa. Trata-se, portanto, de um ambiente cooperativo em que os agentes agem de forma a aumentar a utilidade global do sistema e não a sua utilidade pessoal, além disso existe também uma clara preocupação com o desempenho global da sua equipa em vez do individual.

3.2 Ferramentas de Visualização

Como ferramenta de visualização, o grupo usou as bibliotecas ***JFrame*** e ***JPanel***, de forma a criar uma aplicação gráfica em JAVA que demonstrasse os vários movimentos dos agentes desenvolvidos e o decorrer do próprio jogo.

Foi então desenvolvido um jogo de tanques, em que todos os jogadores se encontram numerados com o seu ID para facilitar a deteção dos movimentos realizados por cada equipa. Além disso, existe duas opções relativas à dimensão do mapa: uma de 15x15 e outra de 35x35. O grupo decidiu acrescentar um mapa de 15x15, uma vez que facilita a deteção de *bugs* durante a implementação do sistema, além de auxiliar na visualização dos movimentos dos vários tanques.

Na seguinte figura encontra-se um excerto de um jogo a decorrer na interface gráfica desenvolvida com um mapa de 15x15.

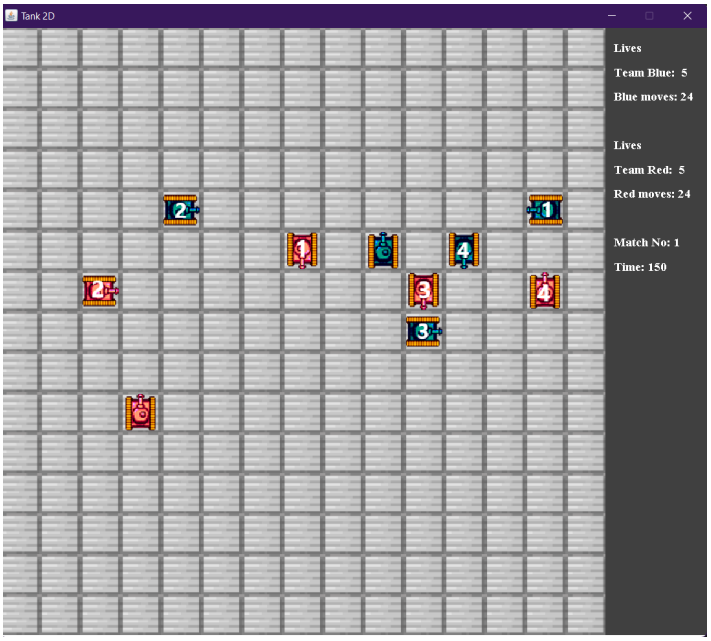

Figura 12: Interface gráfica - Mapa 15x15

É também importante realçar o painel lateral à direita que se encontra na interface gráfica. Este contém várias métricas de performance que o grupo considerou tais como o número de jogadores vivos de cada equipa, o número de movimentos realizados por cada uma e ainda o tempo total de jogo em segundos.

Na figura seguinte está representada a interface gráfica desenvolvida para o momento em que o jogo acaba, anunciando qual a equipa que ganhou, ou seja, aquela que conseguiu eliminar todos os elementos da equipa adversária.

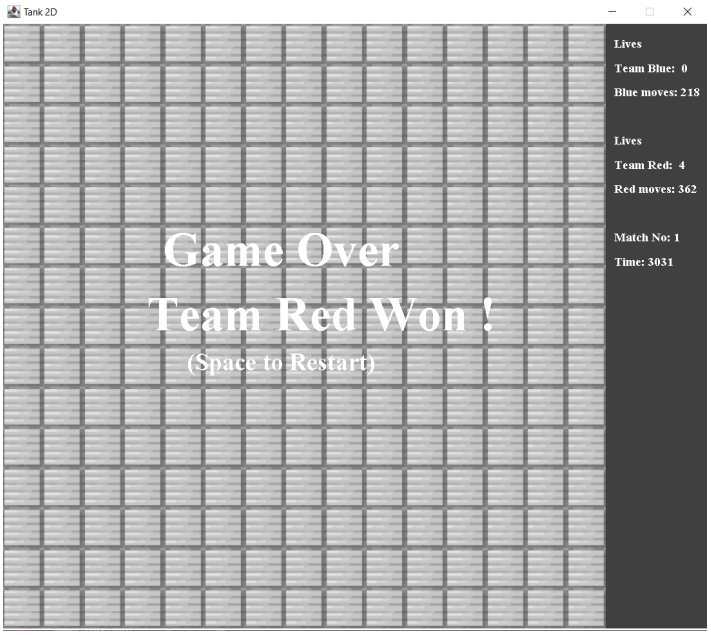

Figura 13: *Game Over* - Interface Gráfica

Já na figura 14, é apresentada a interface gráfica de um jogo quando é escolhido o mapa de dimensão 35x35.

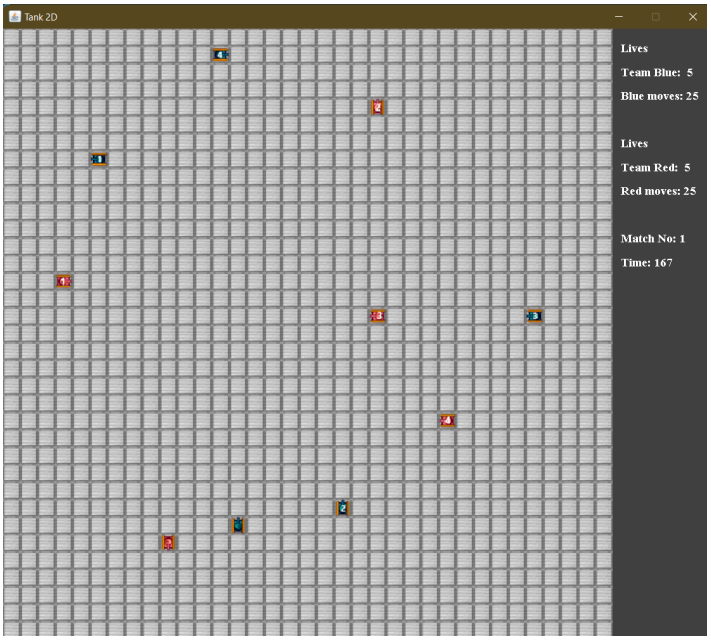

Figura 14: Jogo com mapa 35x35

Além disso, através da opção de visualização da GUI (*Graphical User Interface*) da biblioteca JADE, é possível visualizar os vários containers do sistema desenvolvido, e os agentes inseridos em cada um deles com o seu respetivo id.

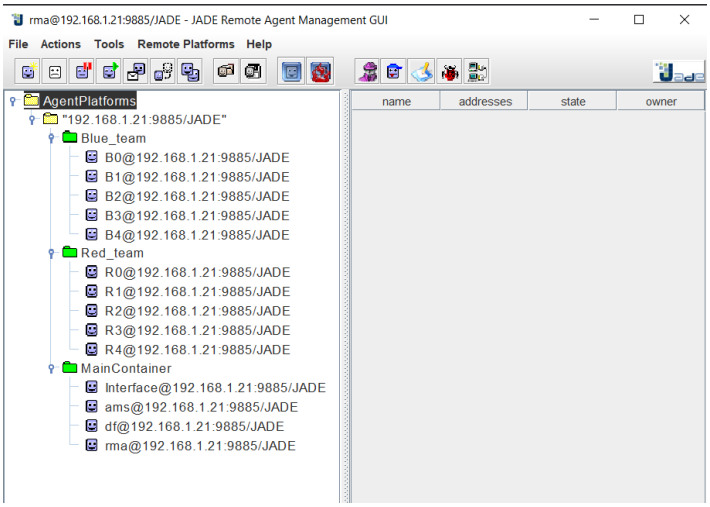

Figura 15: GUI do JADE

3.3 Avaliação e Testes de Performance

Tendo as estratégias dos jogadores implementadas foi necessário avaliar a performance dos mesmos em diferentes cenários. Assim sendo, a primeira situação considerada foi começar sempre com as posições dos jogadores fixas num mapa com dimensões 15x15. De seguida, utilizando o mesmo mapa foram feitos testes definiu-se posições aleatórias. Por fim, considerando ainda posições aleatórias foram colocados os jogadores num mapa com as dimensões pretendidas, isto é, 35x35.

Desta forma, para testar cada cenário executou-se 10 jogos para cada e registou-se os resultados obtidos. Para fazer a análise da performance serão apresentados ao longo deste tópico os resultados de 3 jogos exemplo sendo que no final serão apresentadas algumas estatísticas calculadas utilizando os valores totais (10 jogos).

Para tal, foi necessário estabelecer um conjunto de métricas que o grupo considera que avalia de forma adequada a performance do sistema implementado. Em primeiro lugar, definiu-se o tempo total de jogo, sendo que quanto menor for este tempo, então mais eficientes são as jogadas que os agentes jogadores fazem. De seguida, considerou-se o número total de jogadores vivos em cada equipa. Sempre que um jogo acaba, uma das equipas deve ter 0 jogadores em jogo, enquanto a equipa que ganhou, deverá sempre ter o maior número possível de jogadores vivos. Quando este número é 5 então podemos concluir que a equipa teve um bom desempenho durante o jogo, uma vez que conseguiu ganhar o jogo sem perder nenhum elemento da equipa. Por fim, o grupo ainda contabilizou o número total de movimentos que cada equipa fez num jogo. Uma equipa demonstra um bom desempenho e performance num jogo realizado, quando consegue eliminar todos os elementos da equipa contrária, com o menor número possível de movimentos.

3.3.1 Mapa reduzido (15x15) com posições fixas

Através da análise das tabelas 3, 4 e 5 pode-se verificar, em primeiro lugar, que quando o número de jogadores vivos da equipa vencedora é maior o tempo total de jogo é menor. Esta variação corresponde ao que era esperado, uma vez que, quanto menor a quantidade de jogadores mais difícil será derrotar o adversário. Tal como era expectável, confirma-se que o número de movimentos dos jogadores aumenta com o tempo total de jogo.

Por outro lado, observando os resultados globais na tabela 6 é de notar que a equipa vencedora tem sempre um maior número de movimentos, visto que a partir de um determinado momento terá muito mais jogadores em campo do que o adversário.

Métrica	**Equipa Azul**	**Equipa Vermelha**
Tempo total de jogo (ms)	44 280	44 280
Nº de jogadores vivos	5	0
Nº de movimentos	175	261

Tabela 3: Resultados obtidos no jogo 1 com posições fixas

Métrica	**Equipa Azul**	**Equipa Vermelha**
Tempo total de jogo (ms)	44 660	44 660
Nº de jogadores vivos	0	5
Nº de movimentos	265	120

Tabela 4: Resultados obtidos no jogo 2 com posições fixas

Métrica	**Equipa Azul**	**Equipa Vermelha**
Tempo total de jogo (ms)	65 700	65 700
Nº de jogadores vivos	4	0
Nº de movimentos	302	244

Tabela 5: Resultados obtidos no jogo 3 com posições fixas

Métrica	**Valor**
Média tempos de jogo (ms)	51 710
Média de movimentos da equipa vencedora	275
Média de movimentos da equipa perdedora	183

Tabela 6: Resultados globais para 10 jogos

3.3.2 Mapa reduzido (15x15) com posições aleatórias

Analisando os valores das tabelas 7, 8 e 9 verifica-se que no cenário com posições aleatórias os resultados obtidos são mais variáveis comparativamente ao cenário anterior. Isto significa que houve posições iniciais que favoreceram a existência de mais situações de ataque, ou seja o jogo acabou mais rápido, e outras onde ocorreu o contrário.

Apesar da maior variabilidade, analisando os resultados globais (tabela 10) podemos verificar que, comparativamente ao cenário das posições fixas, os valores obtidos são bastante semelhantes. Isto significa que independentemente da posição onde começam, os jogadores são sempre capazes de atingir o seu objetivo como equipa.

Métrica	**Equipa Azul**	**Equipa Vermelha**
Tempo total de jogo (ms)	47 420	47 420
Nº de jogadores vivos	0	4
Nº de movimentos	130	180

Tabela 7: Resultados obtidos no jogo 1 com posições aleatórias

Métrica	**Equipa Azul**	**Equipa Vermelha**
Tempo total de jogo (ms)	30 400	30 400
Nº de jogadores vivos	5	0
Nº de movimentos	175	89

Tabela 8: Resultados obtidos no jogo 2 com posições aleatórias

Métrica	**Equipa Azul**	**Equipa Vermelha**
Tempo total de jogo (ms)	88 100	88 100
Nº de jogadores vivos	4	0
Nº de movimentos	347	231

Tabela 9: Resultados obtidos jogo 3 com posições aleatórias

Métrica	**Valor**
Média tempos de jogo (ms)	55 416
Média de movimentos da equipa vencedora	232
Média de movimentos da equipa perdedora	142

Tabela 10: Resultados globais para 10 jogos

3.3.3 Mapa real (35x35) com posições aleatórias

Por fim, analisando os resultados para o mapa real (tabelas 11, 12, 13 e 14), verifica-se que comparativamente ao mapa reduzido existe um aumento do tempo de jogo e movimentos, o que era de esperar visto que os jogadores estão mais dispersos e distantes entre si. Comparando com os cenários anteriores, observa-se que este crescimento não foi demasiado significativo, o que significa que, apesar do aumento das dimensões do mapa, o jogo continua a ser competitivo e fluído.

Métrica	Equipa Azul	Equipa Vermelha
Tempo total de jogo (ms)	121 260	121 260
Nº de jogadores vivos	0	5
Nº de movimentos	309	632

Tabela 11: Resultados obtidos no jogo 1 com posições aleatórias (Mapa 35x35)

Métrica	Equipa Azul	Equipa Vermelha
Tempo total de jogo (ms)	85 740	85 740
Nº de jogadores vivos	4	0
Nº de movimentos	334	256

Tabela 12: Resultados obtidos no jogo 2 com posições aleatórias (Mapa 35x35)

Métrica	Equipa Azul	Equipa Vermelha
Tempo total de jogo (ms)	79 680	79 680
Nº de jogadores vivos	0	4
Nº de movimentos	226	357

Tabela 13: Resultados obtidos no jogo 3 com posições aleatórias (Mapa 35x35)

Métrica	Valor
Média tempos de jogo (ms)	101 520
Média de movimentos da equipa vencedora	456
Média de movimentos da equipa perdedora	273

Tabela 14: Resultados globais para 10 jogos

4 Conclusão

A realização deste trabalho prático permitiu aplicar e consolidar diversos conceitos abordados ao longo da UC de Agentes e Sistemas Multiagente, nomeadamente os conceitos de agentes, colaboração, *behaviours* ou Agent UML. Posto isto, iremos agora efetuar uma visão crítica, refletida e ponderada do trabalho realizado.

Por um lado, nos pontos positivos destacamos o facto de se ter desenvolvido uma arquitetura descentralizada e bem definida que potencializa a comunicação e colaboração entre agentes. Além disto, salienta-se também o uso de ferramentas de visualização nomeadamente a criação de uma interface gráfica atrativa que permite obter uma visão mais clara e explicativa do jogo que está a ser executado e das decisões que estão a ser tomadas pelos agentes, sendo que por esta razão possuiu um papel fundamental na deteção de *bugs*. Por fim, achamos que outro aspeto bastante positivo e também o facto de se terem aplicado métricas de avaliação de performance com diversos níveis de complexidade.

Relativamente aos pontos a melhorar, salienta-se o facto de não se ter implementado a tarefa extra relativa às técnicas de *Reinforcement Learning*. Por outro lado, o grupo considera também que uma possível melhoria passaria pela implementação de mais métricas estatísticas ao sistema de avaliação, nomeadamente o número de jogadores que foram eliminados contra uma borda.

Concluindo, consideramos que o balanço do trabalho realizado é positivo, sendo que os aspetos a melhorar podem ser facilmente atingidos num trabalho futuro.

5 Trabalhos Futuros

Tendo-se abordado a arquitetura desenvolvida e todas as decisões tomadas ao longo do projeto, iremos agora explorar alguns pontos que seriam interessantes de implementar futuramente. Em primeiro lugar, a implementação de técnicas de ***Reinforcement Learning*** seria uma adição pertinente ao trabalho, permitindo que os agentes jogadores, através de um sistema de recompensa e punição, aprendam ao longo de várias partidas quais são as jogadas que maximizam a recompensa geral, ou seja, atingir a vitória do jogo.

Além disto, outro aspeto interessante de adicionar ao trabalho seria incluir mais estratégias de jogo através da implementação de novos *behaviours*. Uma possível nova estratégia seria a de *Bait*, na qual um jogador pretende atrair jogadores inimigos para uma zona crítica com o intuito de os eliminar.

Por último, seria também relevante aplicar um algoritmo de jogo diferente para cada uma das equipas, como por exemplo, uma equipa possuir uma estratégia descentralizada e a equipa adversária uma estratégia centralizada (baseada na existência de um *Leader*).

Referências

[1] Tutorials Guides, https://jade.tilab.com/documentation/tutorials-guides/

[2] Câmara A., Silva D., oliveira E., Comparing centralized and decentralized multi-agent approaches to air traffic control, https://web.fe.up.pt/~niadr/PUBLICATIONS/2014/ACamaraESM.pdf

[3] Zamberlan A., Sistemas Multiagentes: Comparativo entre Organização Centralizada e Organização Descentralizada.

[4] Ring N., Communication and Cooperation in Decentralized Multi-Agent Reinforcement Learning, https://agents.inf.ed.ac.uk/blog/master-dissertations/nring_msc2018.pdf

[5] Novais P., Analide C., Agentes Inteligentes. (2006)