Universidade do Minho

Escola de Engenharia

Agentes e Sistemas Multiagente

Mestrado em Engenharia Informática

Grupo 8

Catarina Martins
PG50289

Eduardo Magalhães
PG50352

Laura Rodrigues
PG50542

Maio 2023

Índice

Capítulo 1

Introdução

Sistemas Multiagentes são sistemas que recorrem a Agentes Inteligentes, entidades autónomas capazes de compreender os ambientes em que são inseridos e tomar decisões com base em objetivos definidos. Estes sistemas são oriundos da área de Inteligência Artificial Distribuída e, embora estes sistemas sejam distribuídos pelos agentes, cada agente contribui para o objetivo do sistema como um todo. São diversas as áreas que utilizam Sistemas Multiagente quer para modelação ou simulação de comportamentos.

O trabalho prático proposto para a unidade curricular materializou-se na elaboração de um sistema multiagente para a gestão de partidas e chegadas num aeroporto, utilizando a biblioteca SPADE para o desenvolvimento de agentes.

Para tal, foi necessário conceber e desenvolver uma arquitetura distribuída para gerir esta infraestrutura proposta pelos docentes.

Ao longo deste documento propõe-se uma análise do domínio e da arquitetura desenvolvida neste trabalho prático, bem como uma análise de resultados obtidos e sugestões de melhorias ao sistema desenvolvido.

Capítulo 2

Análise do domínio e Objetivos

Tal como referido anteriormente, o objetivo deste projeto foi desenvolver um sistema multiagente que simulasse partidas e chegadas de aviões num aeroporto.

O domínio tratado permitiu assim fazer a gestão de gares de um aeroporto de modo a que aviões de carga e comerciais pudessem aterrar e/ou descolar de forma organizada e eficiente. Isto só foi possível graças à criação dos principais agentes propostos: a **Control Tower**, o **Hangar Manager** e **Flight Manager** (um agente responsável pela apresentação da informação geral sobre os voos em operação).

A **Control Tower** é responsável por gerir os aviões que pretendem aterrar ou descolar no aeroporto. Quando um avião entra em contacto, esta consulta as pistas livres para o avião realizar a operação que pretende, i.e. aterrar ou descolar. No caso da operaçao ser uma aterragem, se ainda houver gares vagas, a Control Tower comunica com o Hangar Manager de modo a obter informação da gare que se encontra mais próximo de uma das pistas que se encontrem livres.

O **Hangar Manager** é responsável por informar a Torre de Controlo sobre o número inicial de gares de cada tipo existentes e sobre a informação da gare livre mais conveniente para realizar a operação, i.e. mais perto de uma das pistas livres. O Hangar Manager gere assim toda a informação sobre as gares existentes no aeroporto, mudando o estado das mesmas e adicionando e removendo informação sobre o avião conforme as operações realizadas pelo mesmo.

O sistema multiagente também possui um agente responsável por apresentar a informação geral sobre os voos em operação (**Flight Manager**). Este agente agrega as informações dos aviões que estão para descolar ou aterrar, incluindo identificador do voo, companhia aérea, origem, destino, entre outros. Este agente está em constante contacto com a Torre de Controlo para manter as informações atualizadas.

Assim, o domínio a ser tratado refere-se à gestão de operações aeroportuárias, abrangendo a comunicação entre os agentes (Control Tower, Hangar Manager, Flight Manager e Plane) e a coordenação das atividades relacionadas com aviões, pistas e gares do aeroporto.

Como objetivos do sistema definimos a criação de uma infraestrutura de gestão efici-

ente para permitir o controlo das partidas e chegadas de aviões no aeroporto, garantindo a disponibilidade de gares e pistas, além de fornecer informações atualizadas sobre as várias operações a serem realizadas.

Capítulo 3

Análise da Arquitetura proposta

Tal como mencionado anteriormente, o sistema multiagente apresentado pelos docentes propunha a simulação de partidas e chegadas de aviões a um aeroporto.

A arquitetura proposta permite facilitar a comunicação entre os vários agentes. Os agentes podem comunicar entre si de forma autónoma, utilizando mensagens e protocolos específicos (graças aos behaviours). Isso permite uma coordenação eficiente entre os agentes, possibilitando a troca de informações para o funcionamento do sistema.

Pela mesma razão é assim possível atualizar em tempo real as informações: graças à constante comunicação entre os agentes, é possível manter as informações atualizadas em tempo real. Isto tornou-se especialmente relevante para a apresentação das informações dos voos e para a verificação da disponibilidade de gares e pistas. Os agentes podem comunicar entre si de forma imediata quando ocorrerem alterações relevantes, garantindo a precisão dos dados.

No entanto, verificámos que passar todas as informações acerca das gares por parte do Hangar Manager para a Control Tower não seria a melhor e mais eficiente abordagem, por sobrecarregar este agente. Decidimos assim, apenas informar a Control Tower acerca das gares de cada tipo que estão disponíveis, de modo a que a Control Tower também possa fazer a gestão do número de gares ocupadas, diminuindo assim as comunicações entre ambos os agentes. Desta forma, a Control Tower apenas necessita de comunicar com o Hangar Manager quando recebe um pedido de aterragem e existem gares livres, de modo a saber qual a gare mais adequada para o fazer tendo em conta a distância à pista onde vai decorrer a operação e quando informa a mesma que um avião vai descolar.

Capítulo 4

Sistema Multiagente desenvolvido

Neste capítulo será apresentada uma descrição do sistema multiagente desenvolvido, acompanhada de diagramas que permitirão uma melhor compreensão da sua arquitetura e funcionamento. A modelação da arquitetura foi feita com base no problema anteriormente descrito, tendo-se assim tentado idealizar todos os componentes necessários para construção do sistema.

Ao longo deste capítulo serão também analisadas as interações entre os agentes que constituíram o sistema, que garantem a integridade deste.

4.1 Arquitetura

Compreender a arquitetura do sistema tornou-se relevante uma vez que permitiu uma melhor organização do projeto e das estruturas a serem criadas.

4.1.1 Diagrama de classes

O primeiro diagrama estudado foi o diagrama de classes, que nos permitiu melhor compreender o âmbito do problema proposto e organizar agentes e classes extra necessárias para o funcionamento do sistema.

Assim, o nosso Sistema Multiagente é constituído por uma classe Airport que agrega os agentes contituintes do sistema desenvolvido. Tal como se pode observar pela figura seguinte, o nosso sistema é composto por 4 agentes: **Plane**, **Control Tower**, **Hangar Manager** e **Flight Manager**.

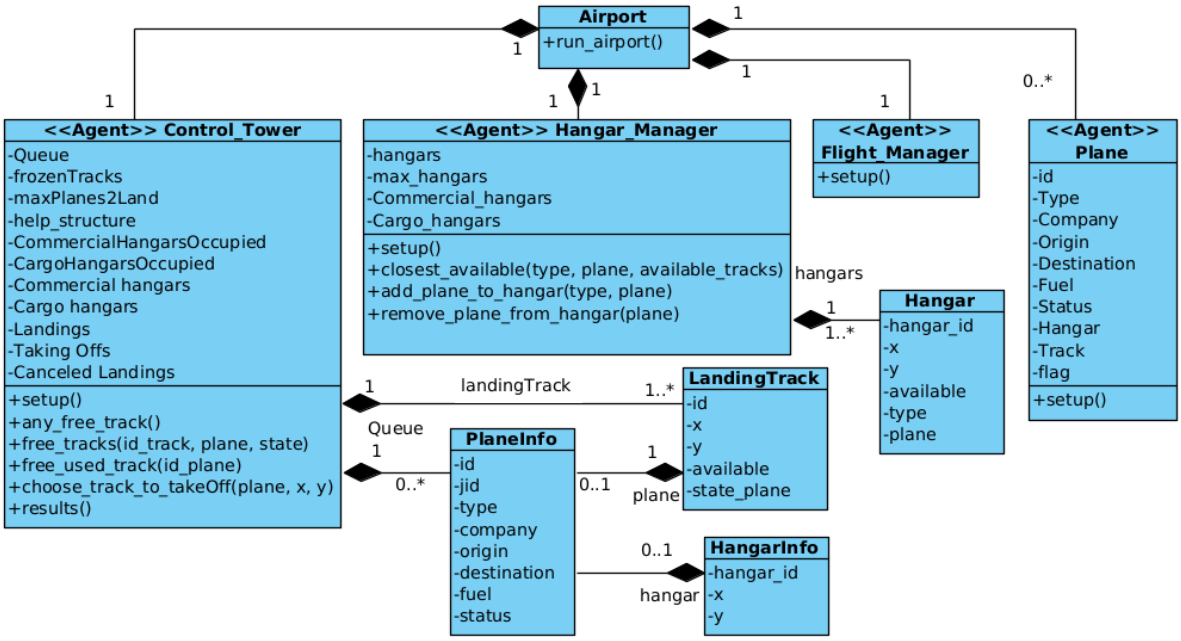

Figura 4.1: Diagrama de Classes

Cada avião comunica com a Control Tower fornecendo-lhe as suas informações:

- ***id***: identificador do avião
- ***Type***: Tipo do avião (se é um comercial ou de transporte de mercadorias)
- ***Company***: Companhia aérea
- ***Origin e Destination***: Países de origem e destino
- ***Fuel***: Quantidade de combustível restante
- ***Status***: Para efeitos de inicialização permite saber se um avião permite aterrar ou descolar
- ***Hangar***: Gare alocado para o estacionamento deste
- ***Track***: Pista de aterragem alocada para a descolagem ou aterragem deste
- ***flag***: Quando um avião pretende descolar e é posto na fila de espera, é definido um TimeOut Behaviour neste agente. De modo a que este behaviour não execute quando o mesmo se encontra em processo de aterragem, é usada esta *flag* de modo a que o behaviour não execute nada

Por sua vez, a Control Tower recebe do Hangar Manager e armazena as informações sobre o número de gares de carga e comerciais disponíveis. Para além disso, é este que regista a lista de aviões em espera (***Queue***) para utilizarem a Pista de Aterragem, fazendo a gestão do número máximo de aviões que podem ficar à espera (número este definido por ***maxPlanes2Land***). Quando o agente Control Tower recebe um pedido

de aterragem verifica quais são as pistas que se encontram livres no momento. Posteriormente, essas pistas são "congeladas", i.e. é definida a sua disponibilidade a False de modo a que seja possível calcular a distância mínima entre as gares e as pistas disponíveis. Para isto, é utilizada a variável ***frozenTracks*** que consiste num array com o id das pistas que foram "congeladas". A disponibilidade da pista é posteriormente restaurada. A variável ***help_structure*** serve para guardar informação dos aviões que fizeram um pedido de aterragem enquanto o processo da mesma se desenrola.

Assim, quando existe um pedido de um avião, é feita a verificação da disponibilidade da pista de aterragem e das gares. Em caso afirmativo, o avião é encaminhado para uma pista (***choose_track_to_takeOff()***) ou para a gare mais próxima disponível (dado calculado pelo Hangar Manager graças à função ***closest_available()***) sendo posteriormente adicionado ou removido da gare. Em caso negativo, o avião poderá ser posto em espera.

Tal como mencionado anteriomente, o Hangar Manager faz a gestão de todas as gares. Para tal, guarda informações sobre estes, entre as quais o número total de hangares, o número de gares comerciais e o número de gares de mercadorias. É este que adiciona e remove os aviões do seu estacionamento.

De modo a ser possível otimizar o funcionamento do Aeroporto foram guardadas as coordenadas X e Y relativas às gares e às pistas de aterragem. Assim, cada avião é encaminhado de forma a percorrer a menor distância possível.

4.1.2 Diagrama de Colaboração

Em seguida foi desenvolvido um diagrama de comunicação (ou diagrama de colaboração) que permitiu melhor compreender como é que os agentes comunicariam e quais os principais *behaviours* que deveriam ser implementados. As comunicações estão ordenadas, à exceção da comunicação identificada pelo número 0 (AskFlightInfo). Decidimos identificar a mesma com o número 0 devido à mesma executar peridodicamente pois a mesma é uma *Periodic Behaviour*.

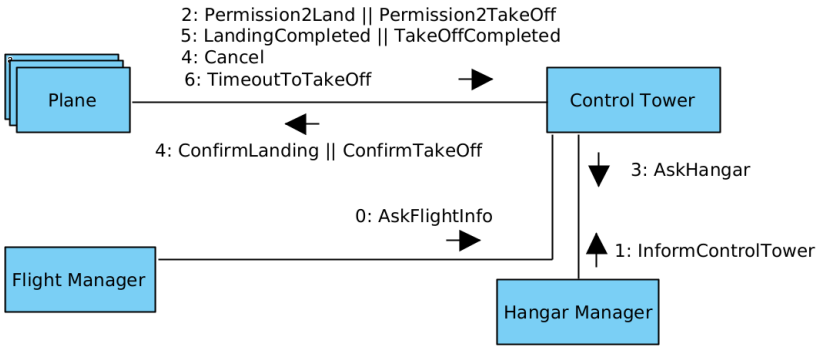

Figura 4.2: Diagrama de Colaboração

Tal como apresentado, como behaviours principais representados no diagrama foi necessário definir 11 tipos de comunicação:

- O Hangar Manager informa a Control Tower do número de gares disponíveis
- Um avião pede permissão à Control Tower para aterrar ou para descolar
- Caso um avião queira aterrar, a Control Tower pergunta ao Hangar Manager por uma gare disponível.
- A Control Tower informa um avião que tem permissão para usar a pista de aterragem e a informação da gare a que se deve dirigir
- Caso esteja em espera hà demasiado tempo, o avião poderá avisar a Control Tower que pretende cancelar o pedido de aterragem
- Uma vez terminada a aterragem ou a descolagem o avião avisa a Control Tower àcerca do sucesso da operação
- Quando um avião já está estacionado há algum tempo numa gare, o mesmo informa a Control Tower que o seu tempo nela terminou e que pretende descolar
- Regularmente o agente Flight Manager pede informações sobre os voos à Control Tower.

4.2 Funcionamento

Para além da compreensão da arquitetura do sistema foi necessário perceber como se processaria o funcionamento do sistema multiagente desenvolvido.

4.2.1 Diagrama de Atividades

Criar um diagrama de atividades permitiu ter uma melhor perceção dos principais fluxos de funcionamento e requisitos das interações entre os diferentes agentes do sistema. Assim, as tarefas de cada agente ficaram bem definidas.

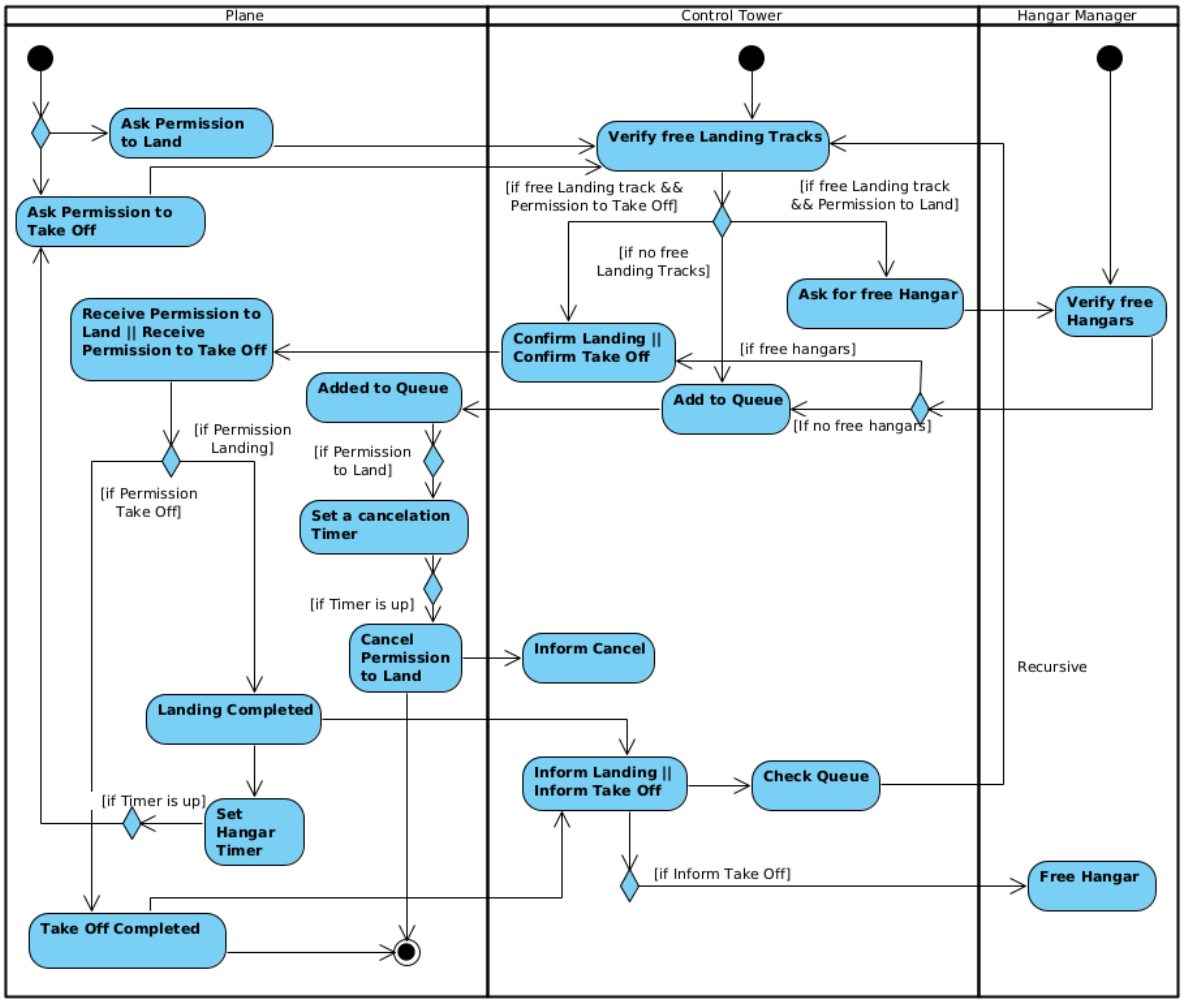

Figura 4.3: Diagrama de Atividades

Na figura anterior apresenta-se o diagrama de atividades concebido. Neste é possível confirmar o que foi desenvolvido anteriormente.

A Control Tower é responsável pela gestão geral do aeroporto, encaminhando os aviões para as diferentes pistas e gares. É também esta que faz a gestão dos aviões que ficam em espera.

As únicas funções do gestor de Gares são: manter a Control Tower informada sobre a disponibilidade de gares e atualizar a sua ocupação no sistema.

O avião, para além de fazer pedidos, é também responsável por gerir dois timers:

- Timer de cancelamento: tal como mencionado anteriormente, quando um avião se encontra há demasiado tempo na fila de espera pode **cancelar o pedido de aterragem**. Para tal, quando o timer definido acabar, a Control Tower é avisada.
- Timer de Gare: Após a aterragem, o avião dirige-se à gare de estacionamento. Nesta fica um determinado tempo até estar pronto para descolar. Assim, quando este timer termina, o avião faz um pedido de descolagem à Control Tower, recomençando o fluxo do diagrama de atividades apresentado.

Por fim, e tendo em atenção a existência do Timer de Gare, pode-se dizer que uma vez iniciada a "vida"de um avião, esta só termina após o take off.

4.2.2 Detalhes de Funcionamento

Detalhes de funcionamento de alguns behaviours e casos em específico

Behaviour	Tipo	Detalhes de Funcionamento
LandingCompleted	TimeoutBehaviour	Definimos um tempo de operação entre o avião e a pista, na pista de 10 segundos no total, 5 para cada operação de modo a facilitar e uniformizar tempos de espera para a simulação
TakeOffCompleted	TimeoutBehaviour	Definimos um tempo de operação entre o avião e a pista, na pista e entre a gare e a pista de 15 segundos no total, 5 para cada operação de modo a facilitar e uniformizar tempos de espera para a simulação
TimeoutToTakeOff	TimeoutBehaviour	Timer de 50 segundos definido quando um avião aterra no aeroporto. Após esses 50 segundos, o avião pede para descolar.
Cancel	TimeoutBehaviour	Quando o tempo de espera (50 segundos) para aterragem é excedido o avião comunica à Torre de Controlo que cancelou a operação.
Add2RandomHangar	OneShotBehaviour	Behaviour executado no início da simulação para atribuir uma gare aos aviões que são definidos para pedir para descolar.
AskFlightInfo	PeriodicBehaviour	O agente Flight Manager pede frequentemente à Control Tower, informação sobre os voos e informação geral do aeroporto.

Tabela 4.1: Alguns Detalhes de Funcionamento

Ficheiro de configuração

De modo a que o setup do Sistema Multiagente fosse o mais cómodo e facilitado possível, definimos um ficheiro de configuração denomidado *"conf.json"* onde se definem as variáveis que contêm informação sobre o XMPP domain name, a password do XMPP, o número inicial de aviões a descolar e aterrar, o número de gares comerciais e de mercadorias, o número máximo de aviões que pretende descolar que se encontram na

fila de espera e o número de pistas de aterragem que o sistema irá possuir.

Um exemplo do conteúdo do ficheiro de configuração é o seguinte:

```
{
"jid": "@xmpp",
"pwd": "xxxxxx",
"nr_landing": 5,
"nr_takeoff": 25,
"Commercial_hangars": 30,
"Cargo_hangars":10,
"maxPlanesWaiting2Land": 50,
"nr_tracks": 4
}
```

Capítulo 5

Resultados obtidos

Sempre que necessário o agente responsável por apresentar as informações do aeroporto (Flight Manager) imprime-as no terminal.

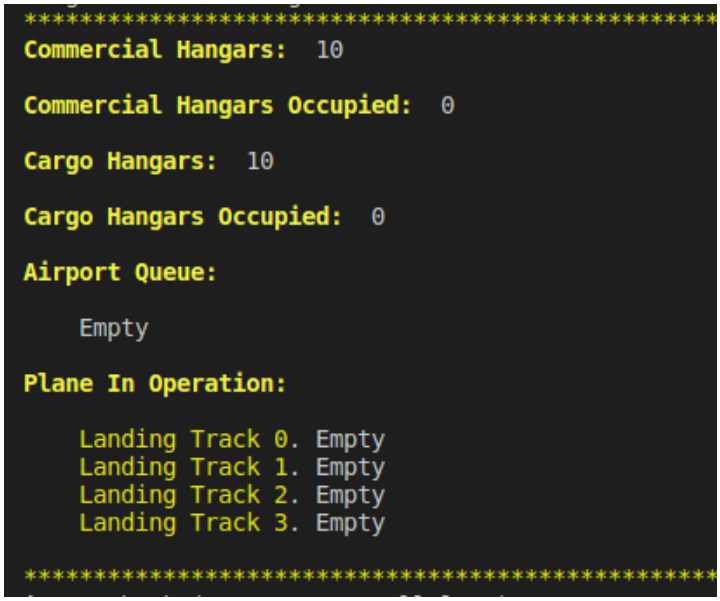

Figura 5.1: Informações sobre os voos ao iniciar o sistema

De 2 em 2 segundos estas informações são atualizadas no terminal com os detalhes dos aviões que chegam pedindo para aterrar.

Como podemos ver pela imagem seguinte, ao chegarem 7 aviões, 4 são distribuídos pelas pistas de aterragem e os restantes ficam em lista de espera. Automáticamente, as gares respetivas (2 comerciais e 2 de mercadorias) são reservadas para estes 4 aviões.

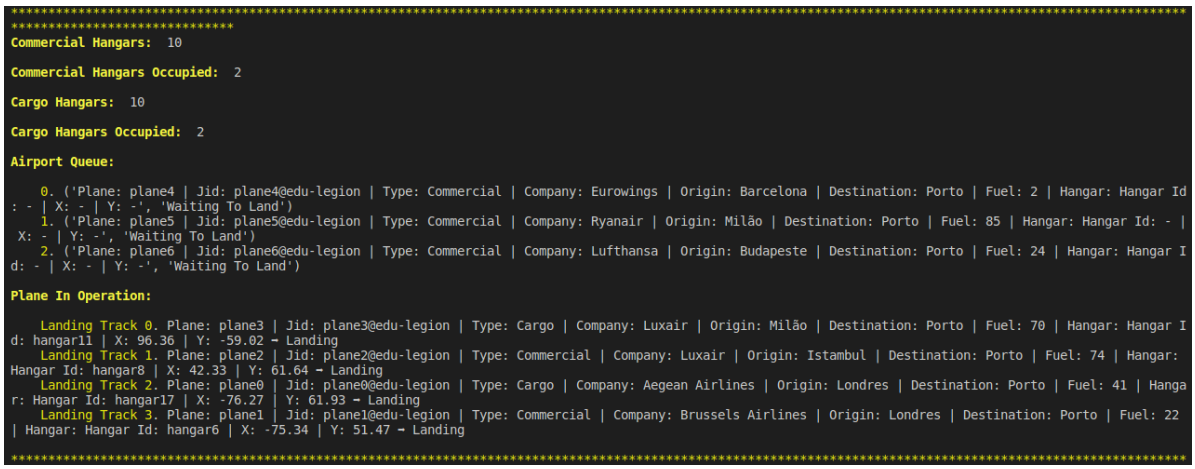

Figura 5.2: Informações - pedidos de aterragem

Quando o timer de todos estes aviões termina, mais uma vez, 4 aviões vão ocupando de cada vez as pistas e os suas gares são libertadas, ficando os restantes na lista de espera.

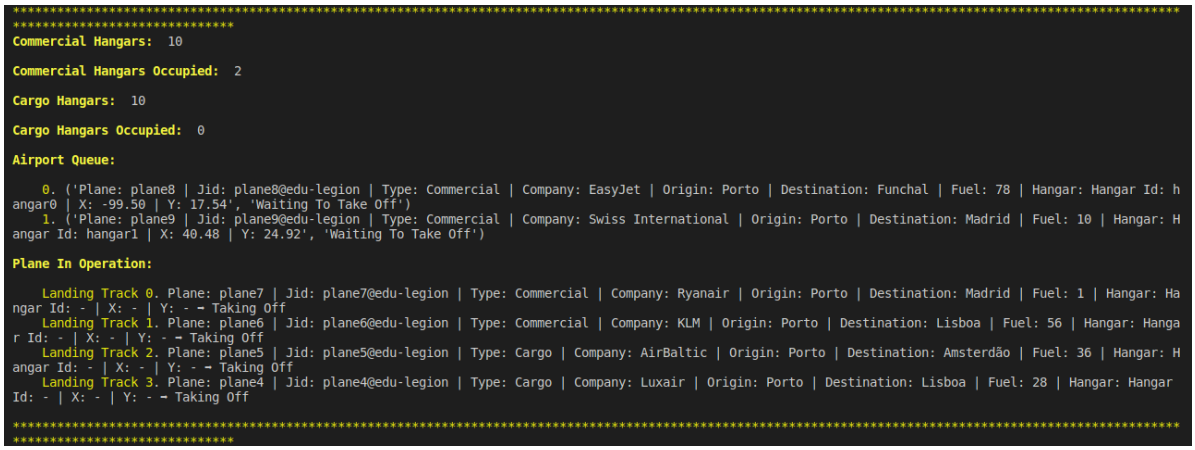

Figura 5.3: Informações - pedidos de descolagem

No final da execução da simulação, é gerado um ficheiro json com informações acerca das aterragens bem sucedidas, descolagens e de aterragens que foram canceladas quer pelo avião, quer pela torre de controlo devido a ter a fila de espera lotada. Como exemplo, apresenta-se o formato deste ficheiro:

```
{"Landings": 0, "Taking Offs": 3, "Canceled Landings": 1}
```

Por fim, com o objetivo de comparar alguns resultados definiu-se o seguinte ficheiro de configuração:

```
{
    "jid": "@xmpp",
    "pwd": "xxxxxx",
    "nr_landing": 5,
    "nr_takeoff": 25,
    "Commercial_hangars": 30,
    "Cargo_hangars":10,
    "maxPlanesWaiting2Land": 50,
    "nr_tracks": X
}
```

Os testes foram realizados alterando o valor de ***X*** - número de pistas de aterragem - para a mesma duração de tempo de teste (5 mins.). Verificou-se que, como seria de esperar, ao aumentar o número de pistas de aterragem, o aeroporto consegue assim dar resposta ao pedido de mais aviões e diminuir o número de Pedidos de Aterragem cancelados, aumentando a taxa de sucesso.

```
{"Landings": 4, "Taking Offs": 14, "Canceled Landings": 47}
```

Teste 1 - sem o método de prioridade - 1 pista

```
{"Landings": 29, "Taking Offs": 50, "Canceled Landings": 32}
```

Teste 2 - sem o método de prioridade - 4 pistas

Analizando os resultados:

$(Landings + TakingOffs)/(Landings + TakingOffs + CanceledLandings)$

Teste	Taxa de sucesso
Teste 1	27%
Teste 2	71%

Tabela 5.1: Taxas de sucesso

Capítulo 6

Sugestões e Recomendações

Tendo em conta os resultados obtidos, o sistema multiagente desenvolvido parece cumprir os objetivos propostos no enunciado deste projeto. Todos os aviões aterram e descolam com sucesso ou cancelam o pedido de aterragem.

No entanto, seria importante salientar possíveis melhorias para o sistema apresentado. Como início de trabalho futuro, tentou-se implementar métodos de negociação: como critério de prioridade de aterragem usar a quantidade de combustível do avião - quanto menos combustível, maior prioridade teria. No entanto, apesar desta variável ***fuel*** ter sido definida (de forma aleatória) não nos foi possível terminar esta implementação.

Destaca-se no entanto, o método de prioridade que foi implementado: quando um avião pretende aterrar, caso haja muitas pistas vazias, este tem prioridade em relação aos que querem descolar.

Outra melhoria poderia ser ter completado a interface gráfica do spade que foi criada mas não foi completada. Apenas a interface gráfica por terminal foi completada com todas as informações e interações necessárias.

Capítulo 7

Conclusões

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*, entre outros.

Por um lado, nos pontos positivos destaca-se o facto de se ter desenvolvido uma arquitetura capaz de dar resposta ao enunciado proposto. No entanto, gostaríamos de ter sido capazes de implementar a funcionalidade extra proposta.

Concluido, considera-se que o balanço do trabalho realizado é positivo, sendo que os aspetos a melhorar podem ser facilmente atingidos num trabalho futuro.