Universidade do Minho
Escola de Engenharia

Universidade do Minho

Mestrado em Engenharia Informática

Agentes e Sistemas Multiagente
2022/2023

Trabalho Prático - Gestão de um Aeroporto

Grupo N.º2

Anabela Lopes Pereira, PG49995

Cláudia Ribeiro, PG49998

Sara Marques, PG47657

Braga, 19 de maio de 2023

Conteúdo

1. Introdução

Partindo do conhecimento adquirido na UC de Agentes e Sistemas Multiagentes, foi nos proposto com este trabalho o desenvolvimento de um sistema multiagente de gestão das partidas e chegadas a um aeroporto com recurso à biblioteca SPADE. Pretende-se que os agentes deste sistema possam comunicar entre si e interpretar as informações recebidas de modo a poderem tomar ações inteligentes de forma coordenada. Nomeadamente, o nosso objetivo é simular a aterragem e descolagem de múltiplos aviões, sem que ocorram conflitos entre estes e sem que nenhum avião seja deixado em espera para aterrar durante alargados períodos de tempo.

Neste relatório, delineamos quais a arquitetura da solução utilizada, e apresentamos os modelos AgentUML que demonstram o seu funcionamento. Descrevemos também detalhes da nossa implementação e das funcionalidades oferecidas pela mesma, e, por último, apresentamos e analisamos os resultados obtidos.

2. Arquitetura e Modelação AgentUML

2.1 Arquitetura do Sistema

Para o trabalho a desenvolver foi proposta uma arquitetura composta por um agente Torre de Controlo (*ControlTower*), sendo este o local onde são centradas as comunicações entre agentes; um agente Gestor de Gares (*StationManager*), que regista e informa sobre o estado de ocupação das pistas e das gares do aeroporto; vários agentes Avião (*Plane*) que efetuam pedidos de aterragem ou descolagem; e um agente responsável por apresentar toda a informação sobre os voos em operação (*Dashboard*).

Para além destes, a nossa arquitetura inclui também um ou mais agentes *Runway*, sendo que cada um comunica exclusivamente com a *ControlTower* e é responsável por processar uma aterragem ou descolagem de um dado avião. Estes processos de aterragem/descolagem incluem esperas (relativas à aterragem/descolagem em si, ou ao movimento entre pistas e gares), logo, caso estes ocorressem na *ControlTower*, dois aviões com pistas e gares livres teriam de esperar que o primeiro terminasse a sua aterragem para o segundo iniciar a sua. Em cenários em que existam mais que uma pista de aterragem/descolagem, esta situação não faz sentido, sendo então necessário que estes processos se realizem antes em agentes *Runway*, permitindo assim a sua execução de modo simultâneo. É de notar que, de modo a não detrair da função centralizadora da *ControlTower*, todas as mensagens geradas pelo processo de aterragem/descolagem na *Runway* serão enviadas para a *ControlTower* e redirecionadas por esta para os agentes relevantes.

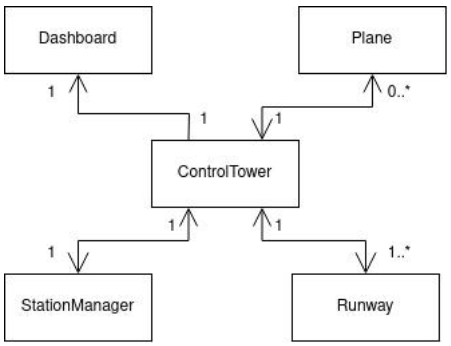

Figura 2.1: Arquitetura de agentes utilizada

Relativamente ao agente *Dashboard* proposto, deve ser apontado que este é apenas responsável por permitir a visualização de informação no terminal, recebendo-a a partir da *ControlTower* e realizando *prints* formatados. Como tal, num programa desta escala, poderia não ser necessário, já que os *prints* poderiam ser realizados imediatamente na *ControlTower* aquando a receção ou envio de pedidos, em lugar de esta ter de comunicar com todo um outro agente. No entanto, caso a informação recebida pelo *Dashboard* fosse mais complexa ou proveniente de várias fontes, já faria sentido a centralização desta funcionalidade num único agente.

2.2 Diagrama de Classes

O programa desenvolvido poderá ser então modelado de acordo com o seguinte diagrama de classes.

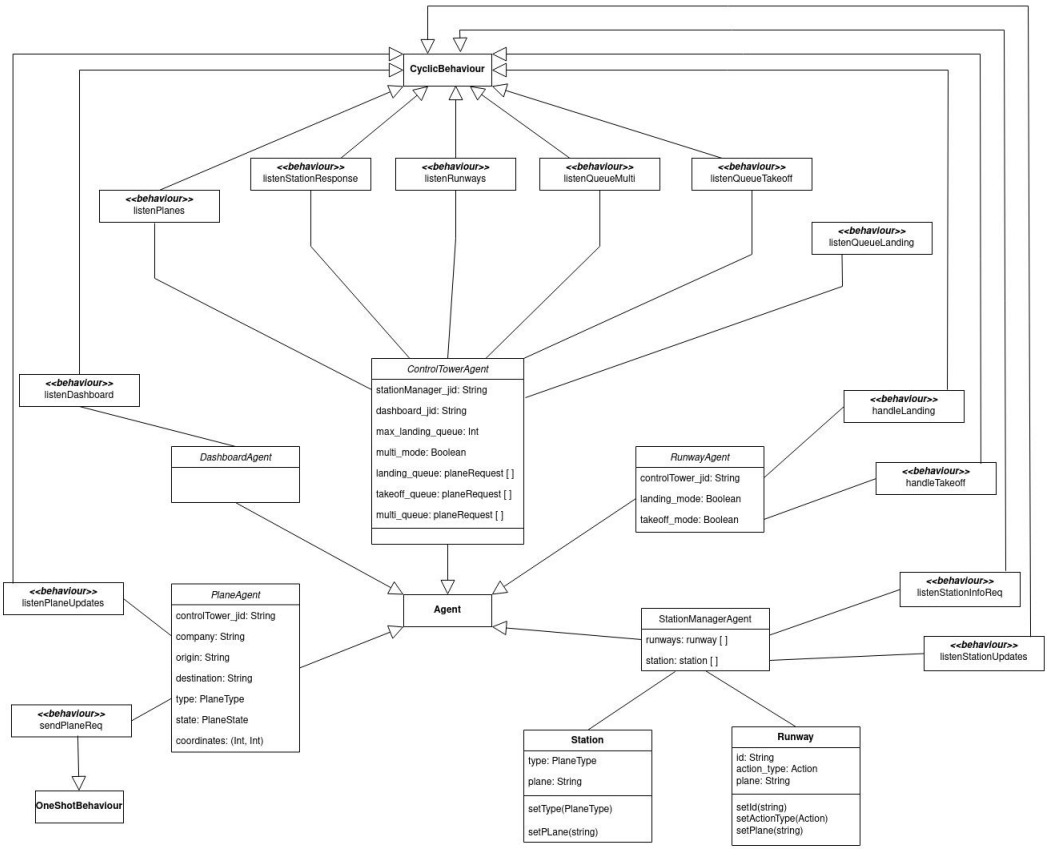

Figura 2.2: Diagrama de Classes - Parte 1

De notar que as classes *Runway* e *Station* são classes que existem apenas para a organização dos dados relativos às pistas e gares no *StationManager*.

Para além destas, existem também três classes Enum, que têm como objetivo limitar o tamanho dos dados incluídos nas mensagens e facilitar a leitura do código.

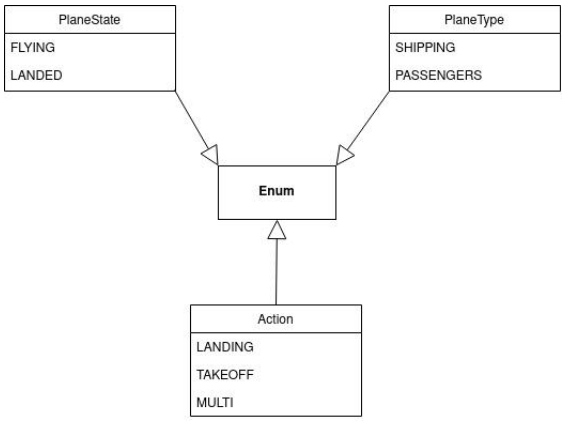

Figura 2.3: Diagrama de Classes - Parte 2

Por último, encontra-se aqui modelada a estrutura dos conteúdos de várias mensagens trocadas entre os agentes em várias interações.

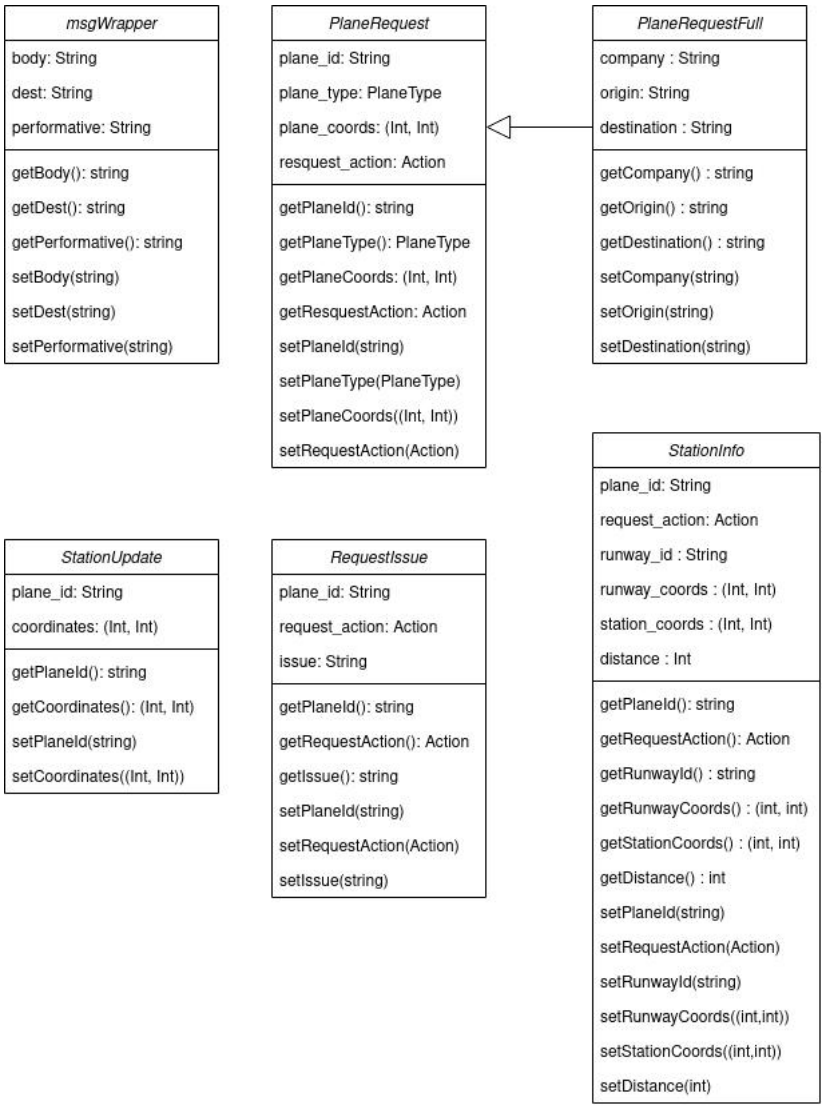

Figura 2.4: Diagrama de Classes - Parte 3

2.3 Diagrama de Colaboração

Em geral, o modo de operação deste sistema multiagente poderá ser modelado através do seguinte diagrama de colaboração.

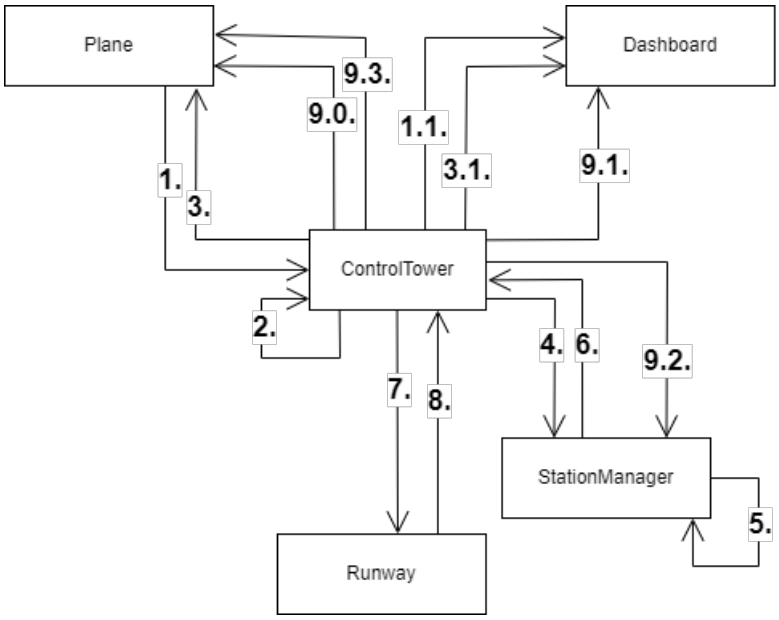

Figura 2.5: Diagrama de Colaboração

As interações no diagrama correspondem à seguinte legenda:

1. Realiza pedido de aterragem/descolagem

1.1. Informa que recebeu um pedido

2. Tenta adicionar pedido à fila

3. Informa que aceitou/recusou o pedido, dependendo se tinha espaço ou não na fila

3.1. Informa que aceitou/recusou o pedido

4. Confirma se o pedido mais à frente na fila pode ser processado

5. Confirma se existe pista/gare livre e apropriada para o pedido

6. Responde com os dados da pista/gare (resposta positiva) ou com recusa (resposta negativa)

7. (Caso receba resposta positiva) Inicia o processo de aterragem/descolagem

8. Devolve várias mensagens relativas ao processo de descolagem/aterragem ao longo do tempo

9.0. (Apenas para aterragens) Informa que vai iniciar aterragem

9.1. Informa que iniciou o processo de aterragem/descolagem, e a progressão do mesmo

9.2. Atualiza estado das gares/pistas

9.3. Atualiza estado do avião

É relevante notar que o agente *ControlTower* poderá ter uma ou duas filas de pedidos, dependendo se existem pistas separadas para aterragens e descolagens (caso esse em que existirá uma fila para cada tipo de pedido) ou se as pistas processam qualquer tipo de pedido (caso esse em que todos os pedidos ficarão numa única fila). Ao mesmo tempo que a *ControlTower* fica à escuta de novos pedidos por parte de agentes *Plane* para adicionar à(s) fila(s), irá também tentar, ciclicamente, pedir ao *StationManager* informações relativas às gares/pista necessárias para a execução da aterragem/descolagem em primeiro lugar na(s) fila(s). Caso obtenha resposta negativa (todas as pistas ou gares apropriadas encontram-se ocupadas) irá repetir o pedido até obter resposta positiva, que irá incluir as coordenadas da gare e pista envolvidas na ação e a distância entre as mesmas (é sempre escolhida a pista mais próxima do avião, e a gare mais próxima da pista, sendo este cálculo realizado pelo próprio *StationManager*). Ao recorrer a uma fila de pedidos, garante-se que é dada prioridade aos aviões baseada na ordem de chegada dos seus pedidos à *ControlTower*.

Após receber resposta positiva da *StationManager*, a *ControlTower* irá redirecionar a informação das gares/pista obtida para o agente *Runway* correspondente, para que a aterragem/descolagem seja processada. Todas as mensagens geradas ao longo desse processo serão enviadas para a *ControlTower* e redirecionadas para os agentes relevantes, sendo este processo de troca de mensagens mais detalhado nos diagramas de sequência na secção seguinte.

2.4 Diagramas de Sequência

De modo a ilustrar a troca de mensagens que ocorre entre os agentes aquando a realização de um pedido de aterragem ou de descolagem, apresentam-se a seguir os diagramas de sequência relativos a cada um desses dois casos.

2.4.1 Aterragem

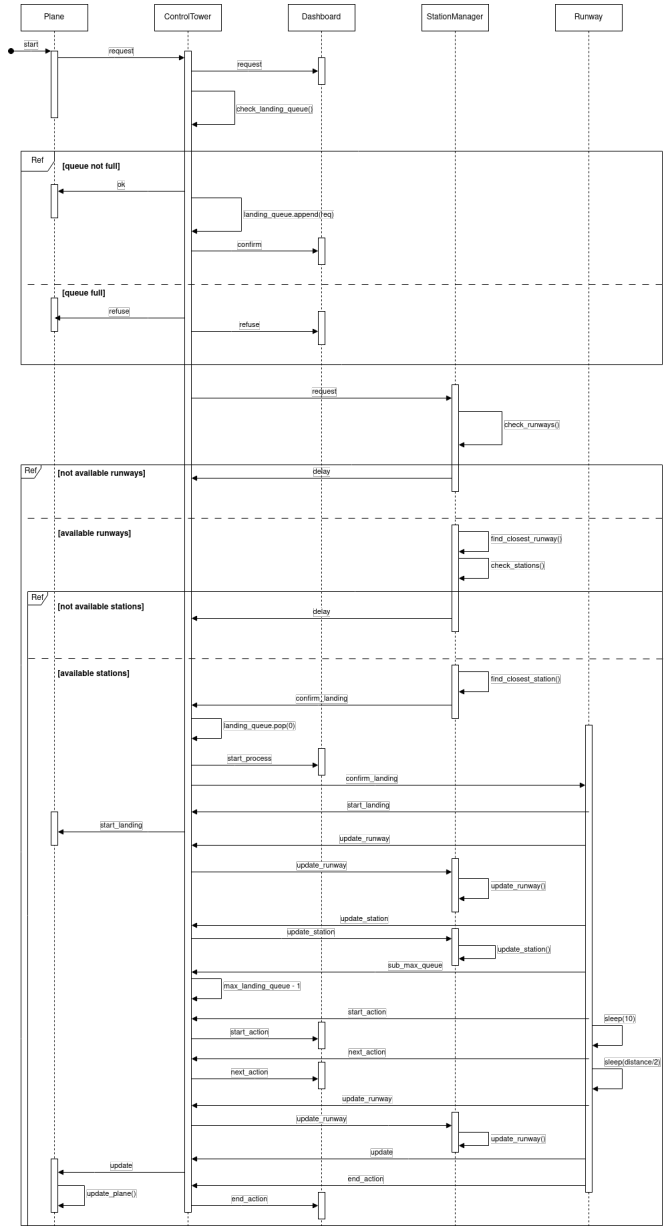

Figura 2.6: Diagrama de Sequência - Aterragem

É de apontar que, 30 segundos após o aterragem do avião, este irá emitir um novo pedido, desta vez de descolagem.

Para além disto, caso o agente *Plane* ainda não tenha recebido a mensagem com o *performative* "ok", que indica que o seu pedido foi recebido, ao fim de um minuto, este irá sair do sistema e procurar um novo aeroporto onde aterrar. Alternativamente, se após receber o "ok"não receber mais nenhuma mensagem ao fim de dois minutos, irá ocorrer um *time out*, e este irá cancelar o seu pedido de aterragem, ser retirado da fila, e sair do sistema para tentar outro aeroporto.

2.4.2 Descolagem

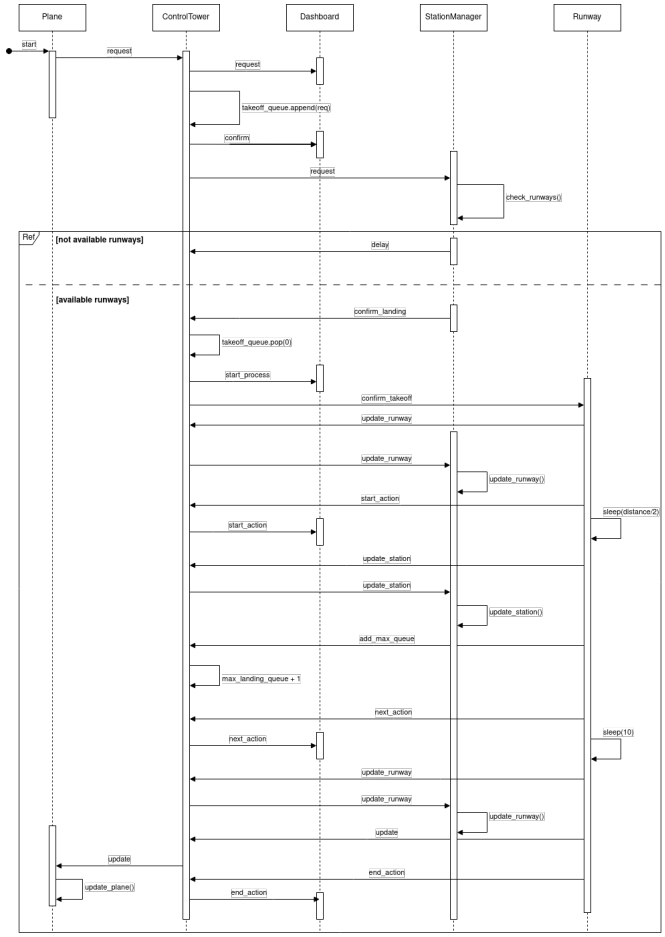

Figura 2.7: Diagrama de Sequência - Descolagem

Após a finalização desta troca de mensagens, o avião que a iniciou irá sair do sistema.

3. Implementação

3.1 Valores da Simulação

O programa desenvolvido é iniciado, por defeito, com uma pista de aterragem, uma pista de descolagem, 4 aviões (2 a aterrar e 2 a descolar) e 6 gares. No entanto, o número de aviões, gares, e pistas pode ser alterado através do uso de opções da linha de comandos.

Metade dos aviões gerados irão emitir pedidos de descolagem e a outra metade de aterragem, sendo aqueles com um id par aviões de passageiros e com um id ímpar de mercadorias. As gares serão igualmente divididas entre ocupadas ou livres consoante o número de aviões que se encontram a realizar pedidos de descolagem (ou seja, que se presumem já estacionados), e alternam entre para passageiros e para mercadorias consoante a paridade do seu id, seguindo a mesma lógica que os aviões. Por último, o número de pistas será dividido entre pistas para aterragem e para descolagem, a não ser que seja selecionado o modo multiusos, caso esse em que todas as pistas poderão ser um local tanto de aterragem como de descolagem.

```
*** Options ***

-h, --help : Show options
-p, --planes [number of planes] : Set number of initial planes. They will be split between landing/taking off,
 and passengers/shipping.
-s, --stations [number of stations] : Set number of stations. They will be split between passengers/shipping,
and automatically filled with the already landed planes.
-r, --runways [number of runways] : Set number of runways. They will be split between for landing/takeoff unle
ss multiuse mode is selected.
-m, --multirunway : Multiuse Mode. All runways can handle both landings and takeoffs.
```

Figura 3.1: Opções da linha de comandos

Após a inicialização do programa, um novo avião que pretenda aterrar será gerado a cada 10 segundos.

3.2 Modo Multiusos

Para além de permitir a personalização do número de aviões, gares, e pistas na simulação do aeroporto, procurou-se disponibilizar também um modo que permitisse que cada pista não servisse apenas exclusivamente para aterragens ou descolagens, mas que pudesse ser utilizada para ambas. Este modo permite, por exemplo, testar uma simulação do aeroporto com uma única pista, e observar a congestão gerada e a sua gestão.

Ao ativar o modo multiusos, apenas é afetado o modo de operação do agente *ControlTower* e dos agentes *Runway*. A *ControlTower* deixará de utilizar duas filas separadas para pedidos de aterragem e descolagem, passando então a guardar todos os pedidos numa única fila de modo a poder preservar a prioridade da ordem de chegada dos pedidos. Por sua vez, os agentes *Runway* irão ser simplesmente todos eles ativos com ambos os *behaviours handleLanding* e *handleTakeoff*, quando normalmente só incluiriam um destes dois comportamentos consoante o tipo atribuído à pista que representam.

4. Resultados

4.1 Múltiplas Gares e Aviões

Para este exemplo, o programa será iniciado com **12 aviões, 12 gares e 2 pistas**, uma de aterragem e outra de descolagem. Isto implica que 6 dos aviões iniciais vão estar em voo, e outros 6 já aterrados. A fila de pedidos irá portanto ficar cheia após a receção de todos os pedidos iniciais, pois existirão 6 gares ocupadas, e 6 com aviões que pretendem aterrar nelas em fila.

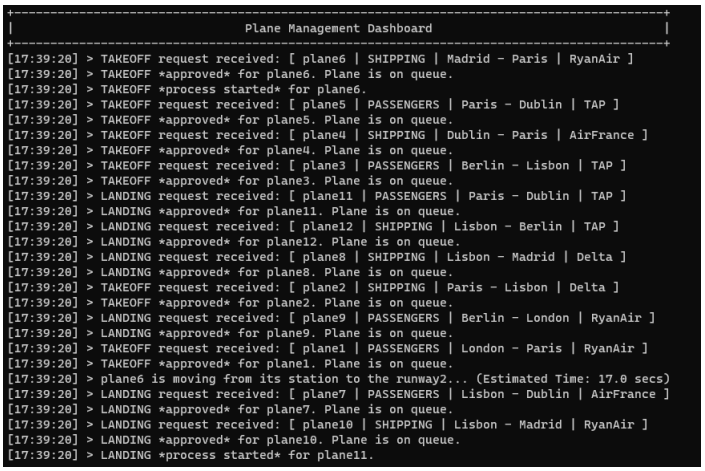

Figura 4.1: Receção dos pedidos iniciais

Podemos observar na imagem seguinte que o próximo pedido de aterragem recebido será recusado, pois não existem gares livres (todas têm ou aviões já estacionados, ou em espera para estacionar). No entanto, quando o plane6 começa a levantar voo a partir da pista, isso implica que a gare de onde saiu já está livre, logo, há um lugar livre e o próximo pedido de aterragem já é aceite e adicionado à fila.

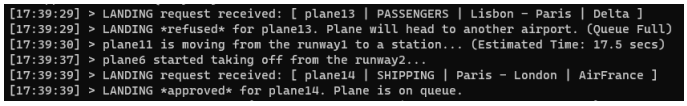

Figura 4.2: Receção de novos pedidos

Por último, é também possível observar neste exemplo um caso de um avião que já estava em fila para aterrar, dar *time out* e sair do sistema para tentar outro aeroporto. Isto ocorre pois, devido ao grande número de gares e limitado número de pistas, a fila de pedidos de aterragem tem um maior limite mas demora mais tempo a que os seus pedidos sejam processados, levando a que os seus últimos possam entrar em *time out*.

Figura 4.3: *Time out* de um pedido de aterragem ao fim de dois minutos em fila

4.2 Múltiplas Pistas

Neste exemplo, o programa foi iniciado com **4 aviões, 6 gares, e 4 pistas**, 2 de aterragem e 2 de descolagem. Como podemos observar, todos os pedidos recebidos (duas aterragens e duas descolagens) começam então a ser processados de imediato nas diferentes pistas.

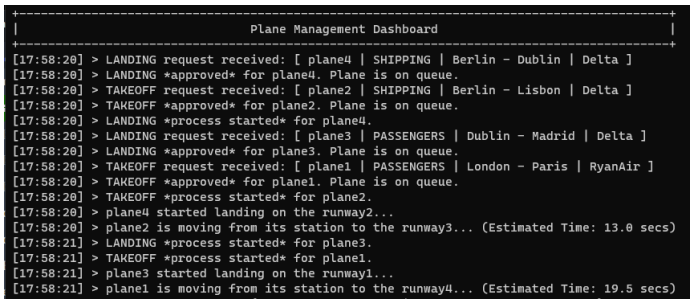

Figura 4.4: Receção e inicio do processamento dos pedidos iniciais

Apesar de se continuarem a receber novos pedidos de aterragem com a chegada de novos aviões, a fila não ficará tão rapidamente congestionada, devido à existência de um maior número de pistas que processem as aterragens e descolagens. No entanto, o limitado número de gares obriga a que o limite de tamanho da fila seja menor, o que levará eventualmente à recusa de pedidos de aterragem caso a fila de pedidos atinja este limite.

4.3 Modo Multiusos

Por último, de modo a demonstrar a funcionalidade multiusos, o programa foi iniciado com **4 aviões, 6 gares e 1 pista multiusos**. Torna-se então possível visualizar como a pista aceita tanto aterragens como descolagens.

Figura 4.5: Início de uma aterragem na pista runway1

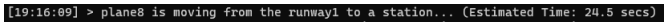

Figura 4.6: Passo intermédio da aterragem

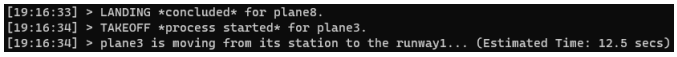

Figura 4.7: Fim da aterragem e início de uma descolagem na mesma pista

Este método pode ser utilizado como um modo de descongestionar a fila de pedidos de modo mais eficiente quando o aeroporto está limitado por um pequeno número de pistas.

5. Conclusão

Consideramos que o trabalho realizado cumpriu todos os requisitos propostos, tendo servido como um método eficaz de consolidar as temáticas abordadas na UC de Agentes e Sistemas Multiagentes ao longo do semestre. Em particular, consideramos benéfica a flexibilidade dada à simulação a gerar, sendo possível personalizar o número de aviões, gares e pistas existentes, e também o modo de operação destas últimas.

Como trabalho futuro, seria interessante registar métricas relativas ao nível de congestão da fila de pedidos e ao número de pedidos recusados e *timed out*, de modo a identificar o número de gares e pistas mais apropriado para aeroportos com diferentes níveis de afluência e que operem em diferentes modos (multiusos ou não). Para além disto, faria sentido implementar um modo de garantir que os aviões de passageiros e mercadorias não compitam entre si por lugares na fila, visto utilizarem gares diferentes, mas continuando a respeitar a prioridade daqueles que comunicam com a torre de controlo primeiro.

Em geral, consideramos o balanço do trabalho realizado positivo, com potencial para expansão em trabalho futuro, mas bastante completo de acordo com os requisitos propostos.