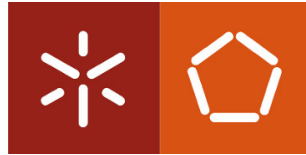


UNIVERSIDADE DO MINHO

ESCOLA DE ENGENHARIA




Agentes e Sistemas Multiagente

Sistemas Inteligentes

Mestrado em Engenharia Informática

Sistema multiagente para a gestão de um aeroporto

PG50345	PG50499	PG50779
		
Duarte Lucas	João Torres	Tiago Ribeiro

Conteúdo

1	Introdução	2
1.1	Caso de estudo	2
1.2	Objetivos	2
1.3	Estrutura do relatório	3
2	Análise à arquitetura sugerida	4
3	Descrição do sistema multiagente	5
3.1	Arquitetura	5
3.1.1	Agentes	5
3.1.2	Comportamentos	6
3.1.3	Performativas	9
3.1.4	Diagrama de classes	10
3.1.5	Diagrama de colaboração	10
3.1.6	Diagrama de atividades	11
3.1.7	Diagrama de sequência	12
3.2	Funcionamento	13
4	Resultados obtidos	15
5	Melhorias para o sistema desenvolvido	17
5.1	Implementação de novos tipos de aviões	17
5.2	Integração com dados meteorológicos	17
5.3	Simulação de falhas	17
5.4	Políticas de prioridade	17
6	Conclusão	19

1. Introdução

Este trabalho prático surge no âmbito da unidade curricular de Agentes e Sistemas Multiagente, pertencente ao perfil de Sistemas Inteligentes do Mestrado em Engenharia Informática da Universidade do Minho.

1.1 Caso de estudo

O caso de estudo deste projeto é a modelação e implementação de um sistema multiagente para a gestão de tráfego aéreo num aeroporto. O sistema é projetado para lidar com diferentes aspetos desta gestão, incluindo a aterragem e descolagem de aviões, a alocação de gares e pistas, e a prevenção de conflitos de tráfego.

O sistema é composto por vários agentes, cada um com sua própria função e responsabilidade. Por exemplo, existem agentes responsáveis pelo controle da torre, agentes responsáveis por gerir as gares e agentes responsáveis pela alocação de pistas. Cada agente possui características específicas, como comportamentos, performativas e habilidades de comunicação, que permitem a execução das suas tarefas de forma eficiente.

O sistema é projetado de forma distribuída, permitindo que cada agente opere independentemente e se comunique com outros agentes em tempo real. A arquitetura do sistema é baseada em agentes, com agentes individuais que interagem uns com os outros para executar as tarefas necessárias. A comunicação entre agentes é feita através da troca de mensagens num protocolo padrão de agentes, permitindo que eles sejam escaláveis e fáceis de implementar em diferentes plataformas.

O caso de estudo deste projeto é relevante porque a gestão de tráfego aéreo é um problema complexo e crítico que exige uma abordagem multifacetada e adaptável. A modelação e implementação de um sistema multiagente permite uma abordagem distribuída e escalável para a gestão de tráfego aéreo, tornando-o mais eficiente, seguro e confiável.

1.2 Objetivos

Os objetivos deste projeto passam por implementar um sistema multiagente para simular o controle de tráfego aéreo num aeroporto, considerando as funcionalidades solicitadas. A biblioteca *SPADE* será utilizada para facilitar a implementação da comunicação entre os agentes, bem como a troca de mensagens.

Para atingir esses objetivos, a primeira etapa será a definição das classes de agentes e dos seus comportamentos, levando em consideração as funcionalidades pedidas. Em seguida, será feita a implementação dos agentes e da comunicação entre eles utilizando a biblioteca *SPADE*. Será necessário também definir um ambiente para a simulação e implementar as interações entre os agentes e o ambiente.

Para garantir a qualidade do projeto, serão realizados testes para validar o correto funcionamento do sistema, verificando se as funcionalidades foram implementadas corretamente e se o comportamento dos agentes está de acordo com as regras definidas. Serão feitas melhorias no projeto a partir dos resultados dos testes, na procura de uma simulação mais realista e eficiente.

do controle de tráfego aéreo.

No final do projeto, espera-se a obtenção de um sistema multiagente funcional, que atenda às funcionalidades requisitadas e permita simular o controle de tráfego aéreo num aeroporto, utilizando a biblioteca *SPADE* como ferramenta de suporte.

1.3 Estrutura do relatório

O presente relatório tem como objetivo apresentar a análise da arquitetura sugerida e a descrição do sistema multiagente desenvolvido para o caso de estudo em questão. Para atingir este objetivo, a estrutura do relatório foi organizada em diferentes seções.

Dito isto, este começa com a introdução, que apresenta o caso de estudo e os objetivos do trabalho. De seguida, no capítulo 2, é feita uma análise da arquitetura sugerida para o sistema multiagente, destacando as suas principais características e vantagens.

No capítulo 3, é apresentada a descrição do sistema multiagente desenvolvido, incluindo a sua arquitetura, agentes, comportamentos, performativas, diagramas de classes, colaboração, atividades e sequência. Este capítulo também aborda o funcionamento do mesmo, explicando como as diferentes partes interagem entre si.

No capítulo 4, são apresentados os resultados obtidos, destacando a sua eficiência e contribuição para o caso de estudo. Já no capítulo 5, são propostas melhorias para o sistema, com o objetivo de aumentar sua eficiência e desempenho. Por fim, no capítulo 6, é apresentada a conclusão do trabalho, ressaltando as principais contribuições e limitações do sistema multiagente desenvolvido.

2. Análise à arquitetura sugerida

Com base na arquitetura proposta, é possível observar uma clara separação de responsabilidades entre os diferentes agentes. Essa divisão permite uma maior modularidade e escalabilidade do sistema, o que pode ser vantajoso para lidar com a complexidade do problema.

A arquitetura distribuída permite que os diferentes agentes trabalhem em conjunto e comuniquem para trocar informações e tomar decisões de forma coordenada. A Torre de Controlo, o Gestor de Gares, o agente agregador de informações de voos e os próprios aviões têm papéis bem definidos e complementares, e trabalham em conjunto para garantir que as informações estejam atualizadas e para que a gestão do aeroporto seja efetuada com segurança e eficiência. O controlo da sincronização das mensagens nesta arquitetura é um fator crítico para garantir a eficiência e a segurança das operações do sistema multiagente. A sincronização das mensagens refere-se à coordenação dos tempos de envio e receção das mensagens entre os diferentes agentes envolvidos.

A arquitetura sugerida tem um grande foco na torre de controlo, pois toda a informação do sistema passa por ela, sendo que atua como o agente líder, em que é o principal responsável pela tomada de decisão do sistema e cooperação entre os vários agentes.

Sendo assim, a distribuição de carga entre os agentes é um fator importante a ser considerado. É possível que alguns agentes, como a torre de controlo, tenham mais trabalho em mãos do que outros. A torre de controlo é responsável por gerir todos os aviões que estão a aterrar e a descolar no aeroporto, o que pode ser uma tarefa muito exigente, especialmente em momentos onde haja um grande tráfego aéreo. Por outro lado, o gestor de gares é responsável por gerir apenas as gares do aeroporto, o que pode resultar numa discrepância nas cargas de trabalho atribuídas a cada agente. Para lidar com esta distribuição de carga desigual, é importante que o sistema seja projetado de forma a permitir que cada agente tenha a capacidade de lidar com sua própria carga de trabalho de maneira eficiente.

A resistência a falhas é outro fator crucial na arquitetura de um sistema multiagente para a gestão de aeroportos. Uma falha em qualquer um dos agentes pode causar atrasos significativos e problemas operacionais, podendo até mesmo levar a acidentes graves. Para garantir a resistência a falhas, a arquitetura precisa de ser projetada com redundância e mecanismos de recuperação em caso de falhas. Por exemplo, uma falha na torre de controlo pode levar a problemas consideráveis, devido à importância deste agente no sistema.

No geral, a arquitetura parece bem estruturada e adequada ao problema em questão, com agentes inteligentes que trabalham em conjunto de forma distribuída para garantir a segurança e a eficiência das operações aéreas. Porém, é necessária a realização de testes para avaliar o desempenho e escalabilidade desta arquitetura.

3. Descrição do sistema multiagente

3.1 Arquitetura

3.1.1 Agentes

Este sistema de gestão de um aeroporto consiste num ambiente cooperativo, no qual os agentes trabalham em conjunto para alcançar objetivos comuns, como garantir a segurança e a eficiência das operações aéreas. O sistema possui vários tipos de agentes inteligentes, cada um com características específicas que contribuem para o bom funcionamento do sistema.

Agente	Descrição
Torre de controlo	Agente reativo, que recebe informações dos aviões e do gestor de gares. Este verifica vários aspetos importantes como a disponibilidade das pistas para aterragem ou descolagem, a distância entre as pistas e as gares, entre outros, para dar instruções aos aviões presentes no aeroporto. Este agente é responsável por tomar decisões imediatas com base nas informações sensoriais presentes no momento.
Gestor de gares	Agente baseado em conhecimento, que realiza a gestão das gares e envia informações sobre as mesmas para a Torre de Controlo. Este agente possui informações sobre as gares, que vão alterando com o passar do tempo, sendo que utiliza estes dados para tomar decisões.
Gestor das informações (Manager)	Agente baseado em comunicação, que recebe informações sobre os diferentes voos e disponibiliza as mesmas para o utilizador do sistema. Este agente comunica com a torre de controlo para obter informações atualizadas sobre os voos.
Avião	Agente reativo, que é capaz de reagir às informações sensoriais e às instruções recebidas da torre de controle para controlar o seu estado enquanto se encontra no aeroporto. Depende das informações recebidas pela torre de controlo para tomar as suas decisões.

Tabela 3.1: Explicação dos vários agentes e das suas caraterísticas

Cada um destes agentes contribui para o sucesso do sistema de gestão do aeroporto, e juntos trabalham de forma cooperativa para garantir a segurança e a eficiência das operações aéreas. O ambiente cooperativo permite que os agentes colaborem entre si, compartilhem informações e trabalhem em conjunto para atingir os seus objetivos comuns.

3.1.2 Comportamentos

Nesta secção do relatório, serão abordados os comportamentos do sistema multiagente, destacando as principais características e funcionalidades dos mesmos. Cada tabela apresentada destaca os comportamentos relativos a cada agente, indicando o tipo destes e o que acontece quando estes comportamentos são executados.

Comportamento	Descrição
gareListenBehav (CyclicBehaviour)	Comportamento que recebe todas as mensagens enviadas para o gestor de gares e que chama os comportamentos necessários para responder aos pedidos recebidos.
gareRequestBehav (OneShotBehaviour)	Recebe dados de um avião, enviados pela torre de controlo, e verifica se existe alguma gare disponível para o tipo deste, enviando a resposta de volta à torre de controlo. Após isto, espera que a torre de controlo envie outra mensagem e ocupa a gare escolhida por ela.
gareLocationBehav (OneShotBehaviour)	Recebe dados de um avião, enviados pela torre de controlo, e liberta a gare em que ele estava enviando a localização da mesma para a torre de controlo.

Tabela 3.2: Comportamentos associados ao gestor de gares

Comportamento	Descrição
managerRequestBehav (PeriodicBehaviour)	Faz o pedido das informações dos vários voos à torre de controlo, recebe a mensagem de resposta e imprime os dados que recebeu.

Tabela 3.3: Comportamentos associados ao gestor de informações

Comportamento	Descrição
landingRequestBehav (OneShotBehaviour)	Comportamento responsável por enviar uma mensagem à torre de controlo para solicitar autorização para aterragem. Recolhe as informações relevantes sobre o avião, codifica-as em formato JSON e cria uma mensagem que é enviada para a torre de controlo.
planeLandingBehav (OneShotBehaviour)	Comportamento responsável por representar o processo de aterragem de um avião. Recebe uma mensagem com informações sobre a pista e a gare onde o avião deve aterrar. Espera o tempo que demora a aterrar e que fica na pista. De seguida envia mensagem à torre de controlo a indicar que a pista está livre. Espera o tempo que o avião demora a mover-se da pista até à gare. Quando chega à gare muda o estado do avião para parked e envia uma mensagem à torre de controlo para remover o avião da lista de aviões que vão aterrar. Depois de esperar 10 segundos é adicionado um novo comportamento de pedido de descolagem.
planeListenBehav (CyclicBehaviour)	Comportamento responsável por receber mensagens da torre de controlo. É cíclico, o que significa está sempre à escuta de novas mensagens (<i>agree_landing</i> , <i>failure_landing</i> , <i>agree_takeoff</i> , <i>failure_takeoff</i> , <i>refuse</i>).
planeTakeoffBehav (OneShotBehaviour)	Representa a descolagem de um avião. Recebe a informação da pista que o mesmo deve usar para descolar, move-o da gare até à pista, aguarda o tempo necessário para a descolagem e, de seguida, informa a torre de controlo que a pista está livre novamente. Além disso, altera o estado do avião para "ar" e termina a sua execução.
planeTimeoutBehav (OneShotBehaviour)	Comportamento responsável por cancelar a espera de aterragem ou descolagem de um avião, no caso de ultrapassar o tempo máximo de espera definido pelo mesmo. Envia uma mensagem para a torre de controlo a indicar que o avião deseja sair da lista de espera e vai para outro aeroporto.
takeoffRequestBehav (OneShotBehaviour)	Comportamento responsável por enviar uma mensagem à torre de controlo a informar que o avião quer descolar. Obtém as informações do voo a partir dos atributos do agente e codifica essas informações. De seguida, cria uma mensagem <i>request_takeoff</i> e envia à torre de controlo.

Tabela 3.4: Comportamentos associados às ações dos aviões

Comportamento	Descrição
towerFreeRunwayBehav (OneShotBehaviour)	Comportamento responsável por gerir a disponibilidade das pistas do aeroporto, atribuindo-as a aviões que estão na lista de espera de aterragem ou descolagem.
towerGiveInformationBehav (OneShotBehaviour)	Comportamento responsável por enviar informações sobre as listas de espera de aterragens e descolagens e sobre os aviões que estão a aterrar ou a levantar, para o gestor do aeroporto. Converte as informações sobre as listas de espera e os aviões num objeto JSON e envia esse objeto numa mensagem para o gestor do aeroporto.
towerLandingBehav (OneShotBehaviour)	Comportamento responsável pelo processo de aterragem de um avião na pista do aeroporto. Se houver gares livres verifica se existe alguma pista disponível para aterragem. Se existir, calcula o caminho mais curto entre a pista e as gares disponíveis e envia uma mensagem de confirmação de aterragem para o avião. Se não houver pistas disponíveis, o avião é adicionado à lista de espera de aterragem e é-lhe enviada uma mensagem a informar que a aterragem não é autorizada e que foi adicionado à lista de espera. Se, por sua vez, a lista de espera de aterragens estiver cheia, é enviada uma mensagem ao avião a informar que a aterragem não é autorizada e que o avião deve encontrar outro aeroporto. Se não houver gares livres, é enviada uma mensagem ao avião a informar que a aterragem não é autorizada e que não há espaço de estacionamento disponível.
towerListenBehav (CyclicBehaviour)	Comportamento responsável por receber as mensagens enviadas pelos outros agentes e encaminhá-las para o comportamento correspondente. Possui várias condições que verificam o tipo de mensagem recebida (pedido de aterragem, pedido de descolagem, pedido de desocupação de pista, pedido de timeout, pedido de informações, notificação de aterragem e notificação de descolagem) e cria um novo comportamento correspondente para lidar com a mensagem recebida. Além disso, também atualiza as listas de aviões a aterrar e a descolar com as informações enviadas pelos comportamentos correspondentes quando um avião completa a operação.
towerTakeoffBehav (OneShotBehaviour)	Comportamento responsável por lidar com o pedido de descolagem de um avião. Se existirem pistas de descolagem disponíveis, envia uma mensagem ao gestor de gares para obter a localização da gare atual do avião. Após receber a resposta do gestor de gares, calcula a pista mais próxima da gare e altera o estado da pista para "ocupada". De seguida, envia uma mensagem de confirmação de descolagem para o avião e remove o mesmo da lista de espera de descolagens. Se não houver pistas de descolagem disponíveis, o comportamento adiciona o avião à lista de espera de descolagens e envia uma mensagem de negação.
towerTimeoutBehav (OneShotBehaviour)	Comportamento executado quando um avião na fila de espera de aterragem atinge o tempo máximo de espera e é cancelado. Recebe o ID do avião que deve ser cancelado e remove-o da lista de espera de aterragem.

Tabela 3.5: Comportamentos associados à torre de controlo

3.1.3 Performativas

As performativas são uma parte fundamental da comunicação em sistemas multiagentes. Performativas são ações ou intenções que um agente pode transmitir a outro agente através da troca de mensagens. As próximas tabelas efetuam uma análise de como estas performativas são utilizadas pelos diferentes agentes para coordenar e cooperar na gestão das operações aéreas do aeroporto.

Direção	Performativa	Descrição
Avião - Torre	request_landing	Indica a intenção de um avião aterrar à torre de controlo
	request_takeoff	Indica a intenção de um avião querer levantar voo à torre de controlo.
	request_free	Avião pede à torre de controlo para desocupar a pista onde aterrou/descolou.
	cancel	Avião indica à torre de controlo que quer cancelar o pedido de aterragem, pois já esperou muito tempo no ar e irá procurar outro aeroporto.
	inform_landing	Avião indica à torre de controlo que já efetuou a aterragem.
	inform_takeoff	Avião indica à torre de controlo que já efetuou a descolagem.
Torre - Avião	agree_landing	Torre de controlo indica ao avião que a aterragem foi autorizada.
	failure_landing	Torre de controlo indica ao avião que a aterragem não foi autorizada.
	agree_takeoff	Torre de controlo indica ao avião que a descolagem foi autorizada.
	failure_takeoff	Torre de controlo indica ao avião que a descolagem não foi autorizada.
	refuse	Indica a impossibilidade de o avião aterrar no aeroporto, pois a fila de espera está cheia ou não existem gares disponíveis.

Tabela 3.6: Performativas relativas à comunicação entre os aviões e torre de controlo

Direção	Performativa	Descrição
Torre - Gare	request	Torre de controlo pede ao gestor de gares a lista de gares disponíveis.
	request_location	Torre de controlo pede ao gestor de gares a localização de uma gare específica.
	request_occupy	Indica ao gestor de gares para ocupar uma certa gare.
Gare - Torre	inform	Gestor de gares disponibiliza a lista de gares disponíveis à torre de controlo.
	inform_location	Gestor de gares fornece a localização da gare especificada pela torre de controlo.

Tabela 3.7: Performativas relativas à comunicação entre a torre de controlo e o gestor de gares

Direção	Performativa	Descrição
Manager - Torre	request_information	Pedido enviado à torre de controlo para requisitar as informações.
Torre - Manager	response_information	Resposta da torre de controlo a enviar as informações.

Tabela 3.8: Performativas relativas à comunicação entre a torre de controlo e o gestor das informações

3.1.4 Diagrama de classes

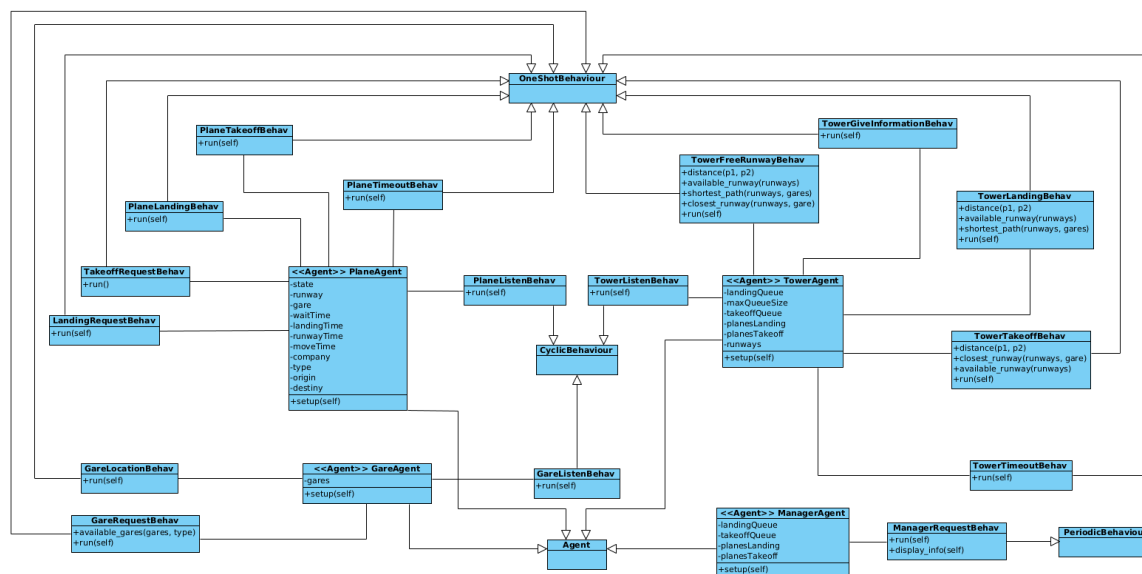


Figura 3.1: Diagrama de classes

3.1.5 Diagrama de colaboração

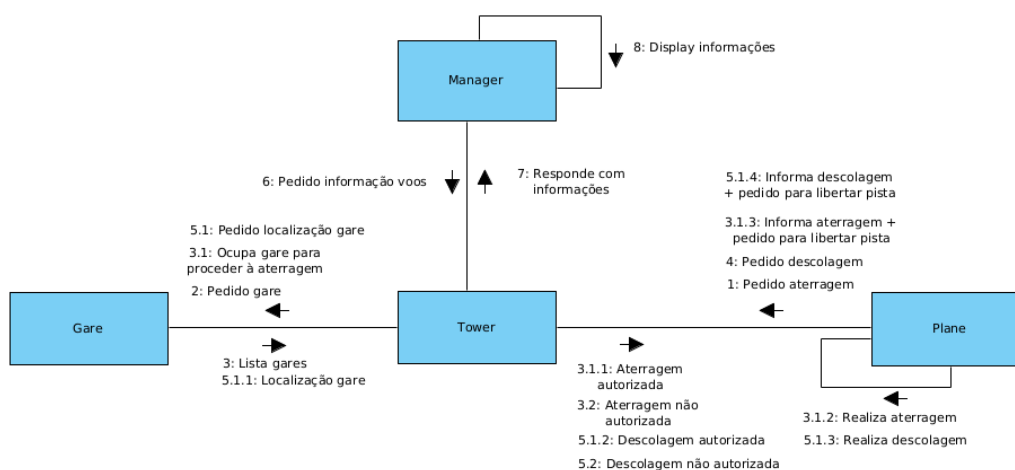


Figura 3.2: Diagrama de colaboração

3.1.6 Diagrama de atividades

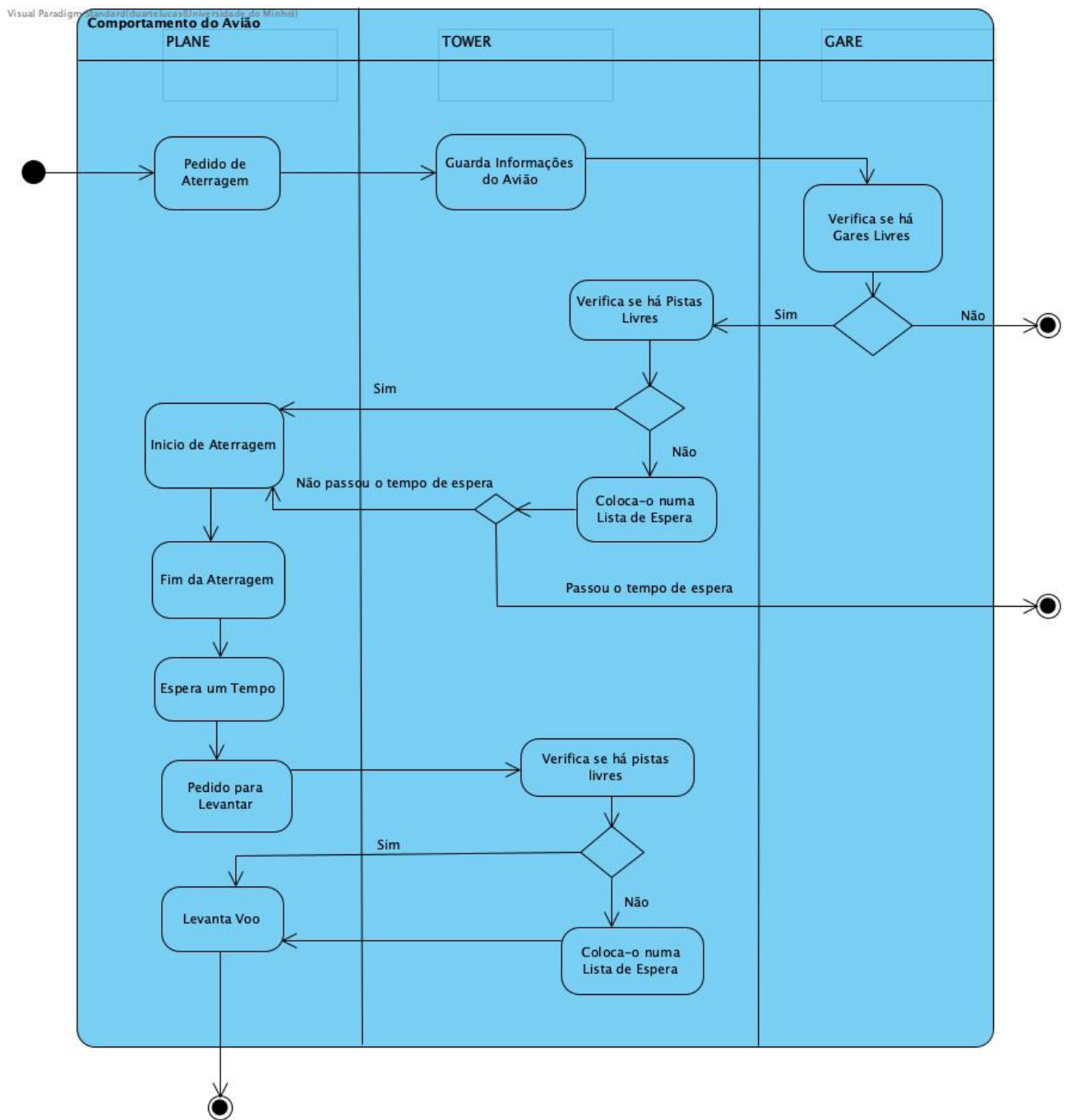


Figura 3.3: Diagrama de atividade

3.1.7 Diagrama de sequência

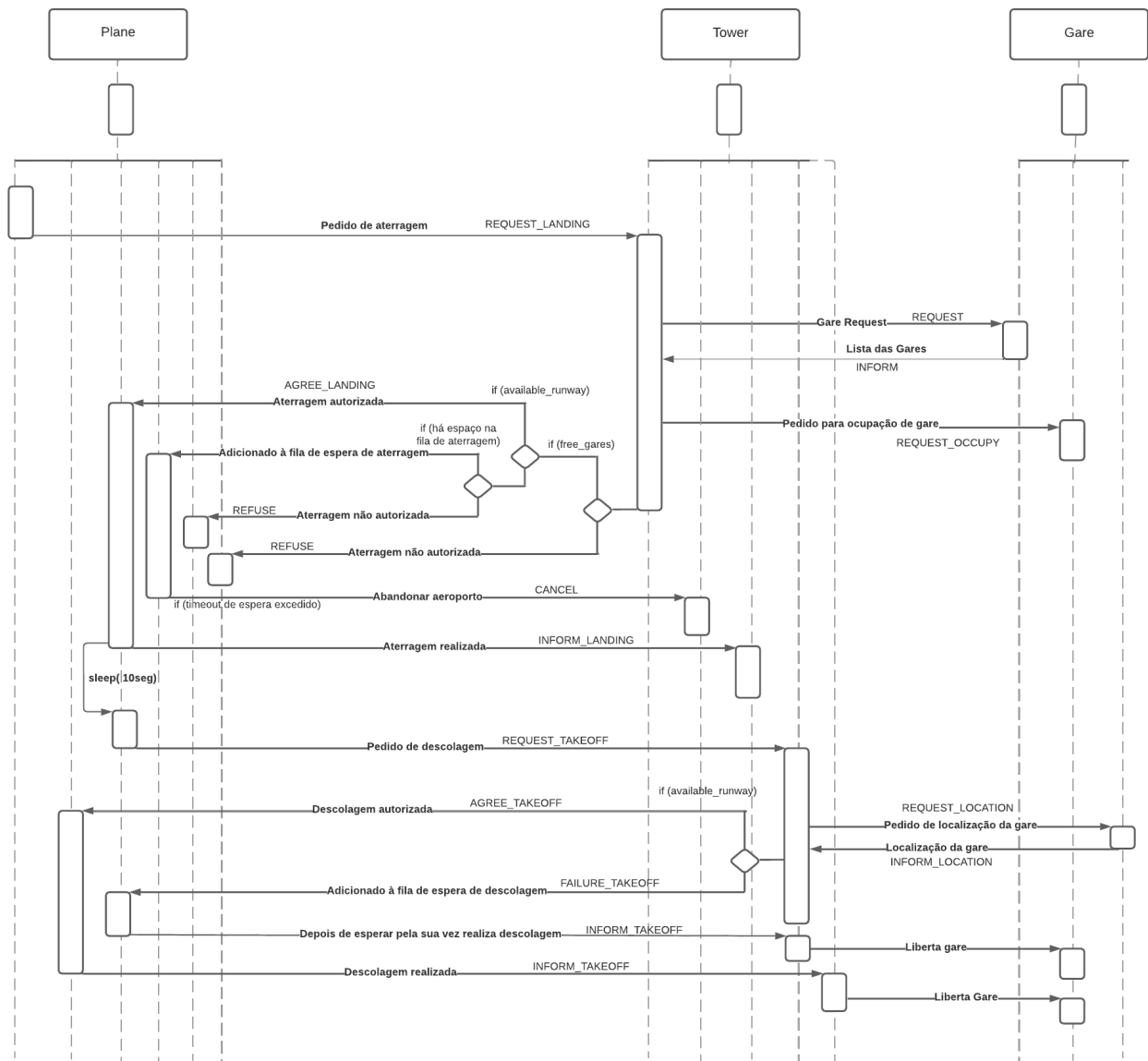


Figura 3.4: Diagrama de sequência

3.2 Funcionamento

O programa desenvolvido tem como objetivo principal gerir e simular as operações de um aeroporto, considerando os processos de aterragem e descolagem de aeronaves. Para atingir esse objetivo, o programa utiliza agentes e comportamentos da biblioteca *SPADE* para gerir e executar as tarefas.

Primeiramente, é realizada a criação e inicialização dos agentes: torre de controlo (TowerAgent), gestor de gares (GareAgent) e o gestor de informações (ManagerAgent). São também criados os aviões iniciais (PlaneAgent), de acordo com o enunciado, ou seja, dois aviões no ar com intenção de aterrar e dois aviões nas gares que pretendem descolar. Os agentes são configurados com os comportamentos necessários para realizar as suas tarefas específicas e, em seguida, iniciados. A partir daqui são criados aviões num período de 10 em 10 segundos, sendo que estes são criados no ar.

De seguida, começam a ser enviados vários pedidos, de diferentes locais, para a torre de controlo. Estes pedidos podem ser:

- Gestor de informações que envia pedidos de informação, num período de 5 em 5 segundos.
- Aviões que estão no ar e enviam pedidos de aterragem;
- Aviões que estão nas gares e enviam pedidos de descolagem;

A torre de controlo verifica as performativas enviadas nas mensagens por cada um destes agentes e invoca os comportamentos necessários para tratar destas solicitações.

Começando pela comunicação mais simples, sendo esta a comunicação entre o gestor de informações e a torre de controlo, este envia num período de 5 segundos a requisitar as informações necessárias para as apresentar. A torre de controlo responde a este pedido, enviando as suas informações de forma codificada, mais especificamente, no formato JSON, utilizando a biblioteca *jsonpickle*, tal como a todas as mensagens enviadas neste sistema multiagente.

Para lidar com a aterragem de um avião, a torre de controlo comunica com o gestor de gares, para verificar se existem gares livres, sendo que o agente responsável pelas gares envia uma lista de gares disponíveis à torre. Caso esta lista esteja vazia, é indicada a impossibilidade de aterrar ao avião e que este terá de procurar outro aeroporto. Caso existam gares livres, a torre de controlo verifica se existem pistas livres. Em caso negativo, o avião é adicionado à lista de espera de aterragens, se esta não estiver cheia. Quando a lista de espera estiver lotada, é pedido ao avião que se reencaminhe para outro aeroporto. Caso existam pistas vazias, é efetuado o cálculo da melhor pista e da melhor gare e é enviada uma mensagem para a aeronave a indicar que pode aterrar, com a indicação da pista e gare a utilizar.

Por outro lado, quando existe um pedido de descolagem, a torre de controlo comunica com o gestor de gares para pedir a localização da gare onde está estacionado o avião que pretende sair do aeroporto. O gestor responde com a localização da gare. Após isto, a torre verifica se existem pistas livres. Caso não existam, o avião vai para a fila de espera de descolagem. Caso contrário, é efetuado o cálculo da melhor pista a utilizar e é dada a autorização para descolar.

Na aterragem ou descolagem de um avião, o sistema tem de esperar que a aeronave efetue a operação em questão, sendo que quando esta deixa de utilizar a pista envia um pedido à torre de controlo para libertar a pista de aterragem utilizada. Quando a aterragem ou descolagem estiver concluída, o avião envia uma mensagem à torre para confirmar que a operação foi efetuada. Durante este processo são alterados os parâmetros do avião como o seu estado (ar,

chão ou estacionado), a gare e a pista utilizadas na operação, os parâmetros da torre de controlo, relativos ao estado das pistas, tal como os parâmetros do gestor de gares, relativos ao estado das gares.

Quando a torre de controlo recebe um pedido para libertar a pista, esta segue as regras e estratégias estabelecidas para garantir a eficiência e segurança das operações de aterragem e descolagem. Sendo assim, a torre percorre a lista de espera de aviões a aterrar, realizando as mesmas operações que quando o avião pede para aterrar (calcular a melhor pista e gare, etc.). Depois de percorrer esta lista, percorre a fila de espera de aviões a descolar, utilizando a mesma lógica. Assim, a prioridade é sempre garantir a aterragem das aeronaves, evitando que sejam forçadas a procurar outros aeroportos para aterrar.

4. Resultados obtidos

Para demonstrar o funcionamento do programa foram feitos testes com agentes de aviões a serem criados de 10 em 10 segundos e de 1 em 1 segundos, sendo que vamos comentar estes testes nesta secção.

```
Plane Agent plane1@tiago-lenovo-ideapad-s145-15iwl starting...
Plane plane1@tiago-lenovo-ideapad-s145-15iwl requesting to takeoff
Plane Agent plane2@tiago-lenovo-ideapad-s145-15iwl starting...
Plane plane2@tiago-lenovo-ideapad-s145-15iwl requesting to takeoff
Plane Agent plane3@tiago-lenovo-ideapad-s145-15iwl starting...
Plane plane3@tiago-lenovo-ideapad-s145-15iwl requesting to land
Plane Agent plane4@tiago-lenovo-ideapad-s145-15iwl starting...
Plane plane4@tiago-lenovo-ideapad-s145-15iwl requesting to land
Plane Agent plane5@tiago-lenovo-ideapad-s145-15iwl starting...
Plane plane5@tiago-lenovo-ideapad-s145-15iwl requesting to land
```

ID	Company	Type	Origin	Destiny	Gare	Runway	State
plane5@tiago-lenovo-ideapad-s145-15iwl	RyanAir	cargo	Lisboa	Braga	None	None	In landing queue
plane3@tiago-lenovo-ideapad-s145-15iwl	Qatar Airways	commercial	Marselha	Braga	g13	r2	Landing
plane4@tiago-lenovo-ideapad-s145-15iwl	Turkish Airlines	commercial	Paris	Braga	g11	r3	Landing
plane1@tiago-lenovo-ideapad-s145-15iwl	Qatar Airways	commercial	Braga	Paris	g1	r1	Taking off
plane2@tiago-lenovo-ideapad-s145-15iwl	Emirates	cargo	Braga	Paris	g2	r4	Taking off

```
Plane Agent plane6@tiago-lenovo-ideapad-s145-15iwl starting...
Plane plane6@tiago-lenovo-ideapad-s145-15iwl requesting to land
```

ID	Company	Type	Origin	Destiny	Gare	Runway	State
plane5@tiago-lenovo-ideapad-s145-15iwl	RyanAir	cargo	Lisboa	Braga	g8	r1	Landing
plane6@tiago-lenovo-ideapad-s145-15iwl	Emirates	cargo	Lisboa	Braga	g12	r2	Landing

```
Plane Agent plane7@tiago-lenovo-ideapad-s145-15iwl starting...
Plane plane7@tiago-lenovo-ideapad-s145-15iwl requesting to land
Plane plane3@tiago-lenovo-ideapad-s145-15iwl requesting to takeoff
Plane plane4@tiago-lenovo-ideapad-s145-15iwl requesting to takeoff
```

ID	Company	Type	Origin	Destiny	Gare	Runway	State
plane7@tiago-lenovo-ideapad-s145-15iwl	EasyJet	cargo	Marselha	Braga	g10	r1	Landing
plane3@tiago-lenovo-ideapad-s145-15iwl	Qatar Airways	commercial	Braga	Barcelona	g13	r2	Taking off
plane4@tiago-lenovo-ideapad-s145-15iwl	Turkish Airlines	commercial	Braga	Porto	g11	r3	Taking off

```
Plane Agent plane8@tiago-lenovo-ideapad-s145-15iwl starting...
Plane plane8@tiago-lenovo-ideapad-s145-15iwl requesting to land
Plane plane5@tiago-lenovo-ideapad-s145-15iwl requesting to takeoff
Plane plane6@tiago-lenovo-ideapad-s145-15iwl requesting to takeoff
```

ID	Company	Type	Origin	Destiny	Gare	Runway	State
plane8@tiago-lenovo-ideapad-s145-15iwl	RyanAir	cargo	Lisboa	Braga	g6	r1	Landing
plane5@tiago-lenovo-ideapad-s145-15iwl	RyanAir	cargo	Braga	Marselha	g8	r2	Taking off
plane6@tiago-lenovo-ideapad-s145-15iwl	Emirates	cargo	Braga	Marselha	g12	r3	Taking off

```
Plane Agent plane9@tiago-lenovo-ideapad-s145-15iwl starting...
Plane plane9@tiago-lenovo-ideapad-s145-15iwl requesting to land
Plane plane7@tiago-lenovo-ideapad-s145-15iwl requesting to takeoff
```

ID	Company	Type	Origin	Destiny	Gare	Runway	State
plane9@tiago-lenovo-ideapad-s145-15iwl	Etihad Airways	commercial	Marselha	Braga	g9	r1	Landing
plane7@tiago-lenovo-ideapad-s145-15iwl	EasyJet	cargo	Braga	Paris	g10	r2	Taking off

Figura 4.1: Execução do programa com aviões a serem criados de 10 em 10 segundos

Com a execução do programa podemos, primeiramente, observar a inicialização dos agentes. Cada agente é instanciado e configurado de acordo com as suas responsabilidades específicas. Este processo envolve a definição das suas características individuais, como estado, tipo, origem e destino (no caso dos aviões), assim como a inicialização das filas de aterragem e descolagem e a preparação para a comunicação e interação com outros agentes.

De seguida, vemos a ativação dos vários pedidos. Estes pedidos podem incluir aviões a solicitar autorização para aterrar ou descolar, o Gestor de Informações a pedir atualizações da Torre de Controlo, entre outros. Cada pedido é gerido de forma assíncrona, o que permite uma operação simultânea e eficiente do sistema como um todo.

Finalmente, podemos observar as informações apresentadas pelo Gestor de Informações. Esta tabela contém detalhes relevantes como o identificador do voo, a companhia aérea, o tipo do voo, a origem e o destino do voo, a pista e a gare atribuídas e o estado do voo (a aterrar, a descolar e em fila de espera). Esta informação é essencial para a gestão e supervisão eficientes das operações do aeroporto, permitindo ter os dados dos aviões organizados de forma simples, concisa e eficiente.

Na figura 4.2 podemos observar a criação de novos aviões a cada segundo e, na coluna do estado, podemos observar aviões em fila de espera tanto de aterragem como de descolagem (na figura anterior não estava representado nenhum exemplo em que um avião estivesse em fila de espera de descolagem). Durante estes testes também é possível observar a atualização de parâmetros como a origem, o destino, a pista e a gare.

ID	Company	Type	Origin	Destiny	Gare	Runway	State
plane25@tiago-lenovo-ideapad-s145-15iwl	Turkish Airlines	commercial	Paris	Braga	None	None	In landing queue
plane26@tiago-lenovo-ideapad-s145-15iwl	Turkish Airlines	cargo	Barcelona	Braga	None	None	In landing queue
plane27@tiago-lenovo-ideapad-s145-15iwl	Turkish Airlines	commercial	Madrid	Braga	None	None	In landing queue
plane28@tiago-lenovo-ideapad-s145-15iwl	EasyJet	cargo	Madrid	Braga	None	None	In landing queue
plane29@tiago-lenovo-ideapad-s145-15iwl	RyanAir	commercial	Madrid	Braga	None	None	In landing queue
plane30@tiago-lenovo-ideapad-s145-15iwl	Qatar Airways	cargo	Barcelona	Braga	None	None	In landing queue
plane31@tiago-lenovo-ideapad-s145-15iwl	EasyJet	commercial	Marselha	Braga	None	None	In landing queue
plane33@tiago-lenovo-ideapad-s145-15iwl	Emirates	cargo	Marselha	Braga	None	None	In landing queue
plane34@tiago-lenovo-ideapad-s145-15iwl	EasyJet	commercial	Lisboa	Braga	None	None	In landing queue
plane16@tiago-lenovo-ideapad-s145-15iwl	Turkish Airlines	cargo	Porto	Braga	g6	r1	Landing
plane17@tiago-lenovo-ideapad-s145-15iwl	EasyJet	cargo	Marselha	Braga	g12	r4	Landing
plane18@tiago-lenovo-ideapad-s145-15iwl	Turkish Airlines	commercial	Porto	Braga	g13	r2	Landing
plane20@tiago-lenovo-ideapad-s145-15iwl	Emirates	cargo	Marselha	Braga	g12	r3	Landing
plane3@tiago-lenovo-ideapad-s145-15iwl	TAP	cargo	Braga	Barcelona	g12	None	In takeoff queue
plane4@tiago-lenovo-ideapad-s145-15iwl	Turkish Airlines	commercial	Braga	Marselha	g13	None	In takeoff queue
plane5@tiago-lenovo-ideapad-s145-15iwl	Turkish Airlines	cargo	Braga	Porto	g8	None	In takeoff queue
plane7@tiago-lenovo-ideapad-s145-15iwl	RyanAir	cargo	Braga	Porto	g9	None	In takeoff queue
plane6@tiago-lenovo-ideapad-s145-15iwl	TAP	commercial	Braga	Madrid	g11	None	In takeoff queue
plane8@tiago-lenovo-ideapad-s145-15iwl	RyanAir	cargo	Braga	Lisboa	g14	None	In takeoff queue

Figura 4.2: Execução do programa com aviões a serem criados a cada segundo

De forma geral, o grupo considera que os resultados obtidos foram bons, visto que o sistema está a cumprir com tudo o que foi pedido no enunciado e consegue recriar o funcionamento da arquitetura de um aeroporto, tal como foi pedido.

5. Melhorias para o sistema desenvolvido

Nesta secção, serão apresentadas algumas sugestões para possíveis melhorias no sistema atual de gestão de um aeroporto. Estas recomendações focam na eficiência, robustez e versatilidade do sistema, sem negligenciar a segurança operacional.

5.1 Implementação de novos tipos de aviões

Atualmente, o sistema lida principalmente com aviões comerciais e de carga. No entanto, uma oportunidade de melhoria poderia ser a implementação de novos tipos de meios de transporte aéreo, como jatos privados. A inclusão do mesmos poderia aumentar a complexidade e a realidade da simulação, permitindo uma maior diversidade de cenários operacionais. Ademais, a gestão de diferentes tipos de aviões poderia requerer a implementação de novas políticas e estratégias de controlo de tráfego aéreo.

5.2 Integração com dados meteorológicos

A segurança das operações aeroportuárias é fortemente influenciada pelas condições meteorológicas. Uma potencial melhoria para o sistema seria integrar dados meteorológicos em tempo real, que poderiam influenciar a decisão da Torre de Controle em termos do número de aeronaves permitidas em espera para aterragem ou descolagem. Por exemplo, em condições meteorológicas adversas, o sistema poderia automaticamente reduzir o limite de aviões em espera para aumentar a segurança das operações.

5.3 Simulação de falhas

Um sistema robusto deve ser capaz de lidar com falhas de maneira eficiente. Atualmente, o sistema pode não estar totalmente preparado para lidar com situações como a falha de um agente ou o encerramento de uma pista de pouso. Uma melhoria sugerida seria a implementação de mecanismos que permitam ao sistema responder de maneira robusta a esses eventos. Por exemplo, em caso de falha de um agente, um sistema de backup poderia assumir as funções do mesmo. Ou, se uma pista for fechada, o sistema poderia redistribuir automaticamente os aviões para as pistas restantes.

5.4 Políticas de prioridade

Por fim, as políticas de prioridade usadas para determinar a ordem de aterragem e partidas poderiam ser otimizadas. Atualmente, o sistema dá uma grande prioridade a aviões que pretendam efetuar uma aterragem, o que pode resultar com que muitos aviões fiquem bloqueados durante muito tempo na fila de espera de descolagem. Outro aspeto a implementar no sistema

poderia ser a adição de aviões com necessidades especiais, como ambulâncias aéreas ou aviões com problemas técnicos. Seria útil implementar políticas de prioridade que permitam que os mesmos tenham prioridade na aterragem ou descolagem.

6. Conclusão

Nesta fase final do projeto constatamos que conseguimos cumprir com tudo o que nos foi pedido. Consideramos que foi um projeto bastante interessante e desafiante.

O sistema mostrou-se capaz de lidar com a complexidade inerente à gestão de um aeroporto, incluindo a coordenação de aterragens e partidas, a gestão de filas de espera e a alocação de gares. Contudo, reconhecemos que sempre há espaço para melhorias e inovação.

Sugerimos algumas melhorias potenciais para o sistema, incluindo a implementação de novos tipos de aviões, a integração de dados meteorológicos em tempo real, a simulação de falhas e a implementação de políticas de prioridade refinadas. Acreditamos que essas melhorias poderiam aumentar a robustez, a eficiência e a realidade da simulação, tornando o sistema ainda mais valioso como ferramenta de gestão e simulação de operações aeroportuárias.

Em suma, este projeto ofereceu uma boa oportunidade para aplicar e aprofundar o nosso conhecimento de sistemas multiagentes e a sua aplicação. Deste modo, consolidamos o conhecimento adquirido nas aulas práticas e teóricas.

Concluindo, este projeto contribui para expandir e por em prática os conceitos adquiridos durante as aulas nas áreas de agentes inteligentes e da utilização de bibliotecas de comunicação de agentes como o *SPADE*. Também nos ajudou a desenvolver novas aptidões e a consolidar toda a matéria lecionada nas aulas práticas e teóricas. Esperamos que o trabalho em si e as melhorias propostas possam contribuir para o desenvolvimento futuro do sistema e para a área de agentes e sistemas multiagentes como um todo.