Relatório 1º fase Agentes Inteligentes

Diogo Rodrigues
orcidIDPG37150, Hugo Gonçalves[PG38928], and Luis Bouça[PG38933]

Universidade do Minho, Departamento de Informática, Largo do Paço 4704-553,
Braga gcii@reitoria.uminho.pt
https://www.uminho.pt/

Abstract. O objetivo com a realização deste trabalho passa por criar um sistema multi-agente com o propósito de simular como funcionaria a gestão do fluxo de tráfego aéreo. Através do recurso de diagramas, mais precisamente, diagramas de sequência, diagramas de estado e diagrama de classes fizemos uma planificação da nossa arquitetura multi-agente e como é que os diversos agentes se irão comportar no ambiente.

Keywords: Agente · Sistema Multi-Agente · Controlo fluxo aéreo.

1 Introdução

Este trabalho consistem em construir um sistema multi-agente de controlo do fluxo aéreo entre vários aeroportos e com múltiplos aviões.

Na figura 1 podemos ver algumas das interações que podem ser efetuadas entre os dois atores principais, Avião e Aeroporto, como o processo de aterragem no canto inferior esquerdo,e a negociação entre aviões com rota de colisão no canto inferior direito da figura, assim como o processo de descolagem no canto inferior direito da figura.

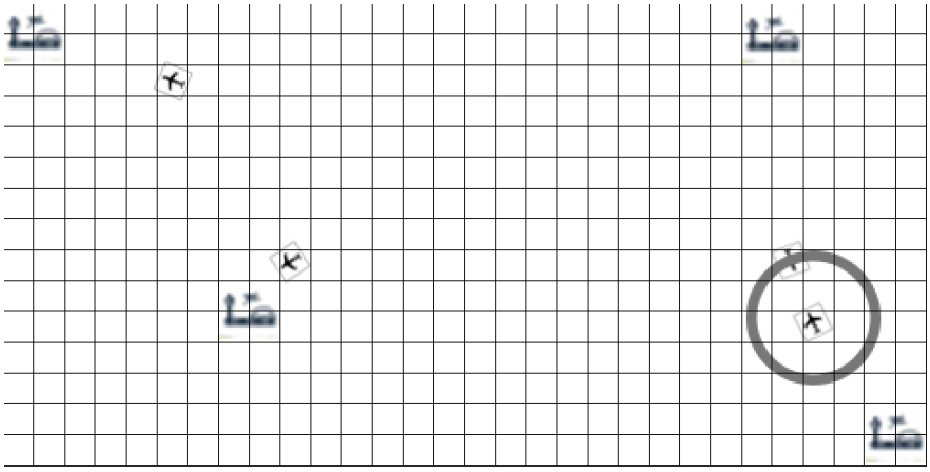

Fig. 1. Exemplo visual do objetivo do trabalho

2 Diagramas de Sequência

2.1 Descolagem

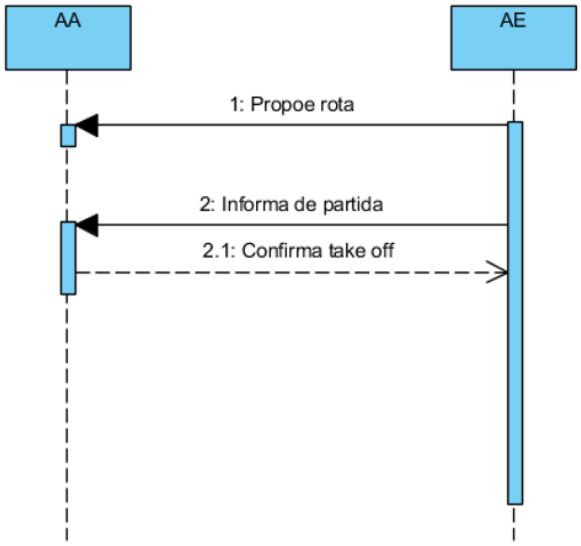

Fig. 2. Diagrama de sequência da descolagem

Quando o programa é inicializado, todos os aviões presentes no sistema irão estar alocados a um determinado aeroporto. No inicio o aeroporto irá fornecer aos aviões uma rota de destino e por conseguinte irá ser adicionado a uma lista de espera para descolar. Esta lista irá sempre ter uma prioridade.

Quando o avião puder descolar o aeroporto envia uma mensagem de confirmação para o avião puder descolar. O avião por sua vez responde ao aeroporto quando tiver acabado de descolar. No final da descolagem o avião é retirado da lista para descolar.

Table 1. Lista de performatives usados.

Tipo de performative	Uso
Inform	**Todas as mensagens irão ser apenas informativas**

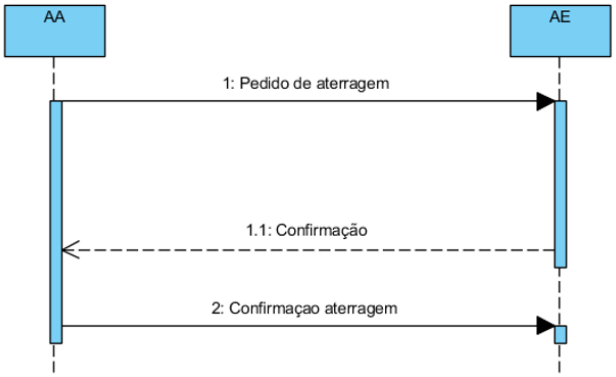

Fig. 3. Diagrama de sequência da aterragem

2.2 Aterragem

Quando o avião se encontra preparado para aterrar este envia um pedido ao aeroporto a requisitar autorização para aterrar. O aeroporto vai de acordo com as suas características como o número de pistas livres, a capacidade máxima de aviões que o aeroporto pode albergar b

No final quando o avião aterra no aeroporto este envia uma mensagem a informar que aterrou e é posteriormente adicionado a uma lista de espera para voltar a descolar com um novo destino fornecido pelo aeroporto.

Table 2. Lista de performatives usados.

Tipo de performative	Uso
Propose	**Quando o avião faz o pedido para aterrar**
Accept Proposal	**Quando o aeroporto responde ao pedido do avião**
Inform	**Quando o avião informa o aeroporto que aterrou**

2.3 Voo

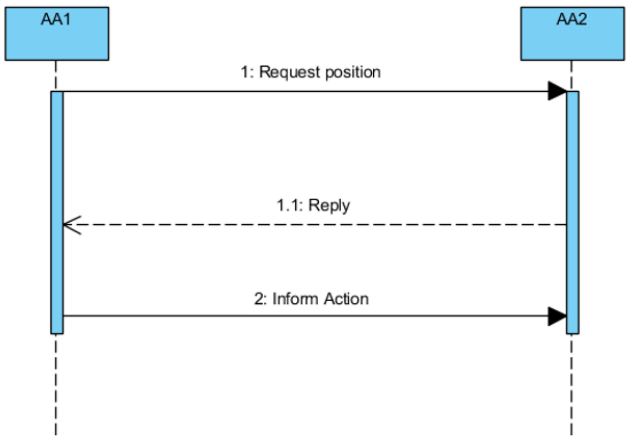

Fig. 4. Diagrama de sequência do voo

Quando o avião se encontra no ar, ele vai estar em constante comunicação com todos os aviões que também estejam a percorrer alguma rota. Para isto o avião envia um pedido de coordenadas em que os outros aviões ao redor dele se encontram. Os aviões que recebem este pedido enviam de volta as suas coordenadas.

Agora caso haja perigo de colisão irá haver uma negociação entre os aviões envolvidos de maneira a verificar qual é a melhor decisão a ser tomada que pode ir desde uma simples redução da velocidade como ser preciso um desvio na rota para evitar o embate

Table 3. Lista de performatives usados.

Tipo de performative	Uso
Request	**Pedido das coordenadas por parte do avião**
Inform	**Resposta com as coordenadas dos respetivos aviões e qual a decisão a ser tomada**

2.4 Interface

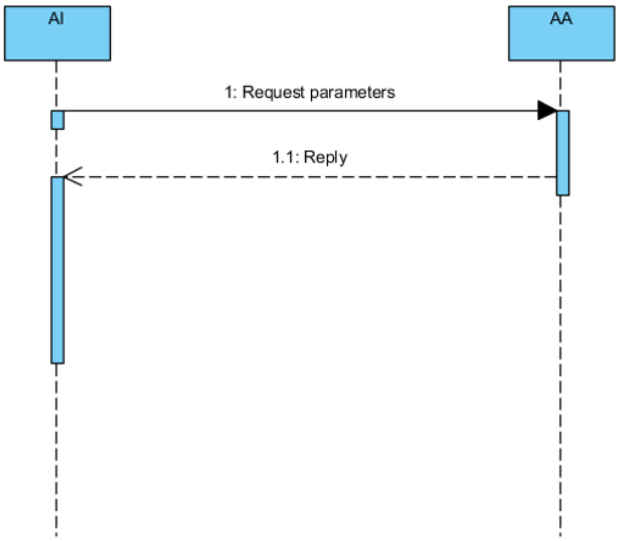

Fig. 5. Diagrama de sequência do voo

Como é dito no enunciado, teremos que ter um agente que esteja em constante contacto com as aeronaves de modo a saber qual o status, tomada de decisões, no fundo as caracteristicas de cada aeronave. Por isso de x em x segundos o nosso agente Interface irá enviar uma mensagem a requerer a informação de cada aeronave. Quando o avião recebe o pedido envia como resposta toda sua informação relativa ao voo, caracteristicas da nave, etc.

Table 4. Lista de performatives usados.

Tipo de performative	Uso
Request	**Pedido dos parâmetros dos aviões feito pelo agente Interface**
Inform	**Resposta com os parâmetros dos aviões**

3 Diagrama de Estados

3.1 Avião

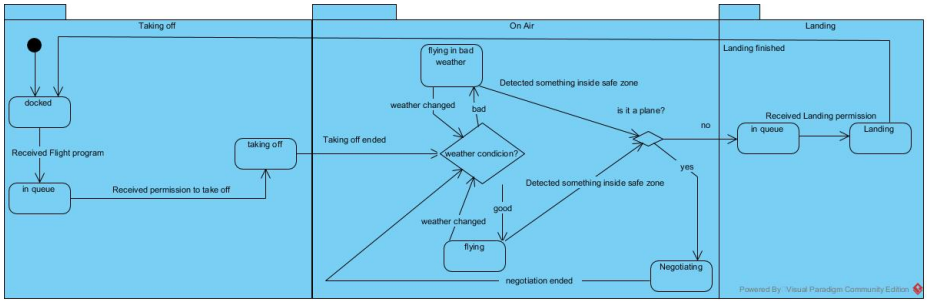

Fig. 6. Diagrama de estados de cada avião

Quando um avião é criado este começa por se encontrar no estado de "em armazém" e assim que este receber o plano de voo muda para o estado de "na fila", no qual permanece até receber a permissão do aeroporto para descolar, o que muda o estado do avião para "em descolagem".

Assim que o avião acabar a descolagem este tem duas opções dependentes do tempo da localização em que ele se encontrar, caso esteja bom tempo este fica com o estado de "a voar", caso não esteja bom tempo, o estado deste torna-se "a voar em mau tempo". Tanto o estado "a voar" como o "a voar em mau tempo" tem as mesmas transições por isso a partir deste momento esse conjunto de estados vai ser definido por conjunto de voo. Se houver uma mudança de meteorologia do espaço que o avião ocupa o estado deste pode ou não alterar para um dos presentes no conjunto de voo.

No momento em que o avião detetar uma presença na sua zona segura existem duas possibilidades, essa presença tanto pode ser um outro avião, o que muda o estado do avião para "negociação", e quando a negociação acabar o estado volta para um dos do conjunto de voo, como a presença pode ser uma aeronave e então o estado do avião torna-se "em fila". Quando o avião receber permissão para aterrar este fica com o estado "a aterrar", e ao acabar a aterragem este volta para o estado de "em armazém" e o ciclo volta ao inicio.

3.2 Aeroporto

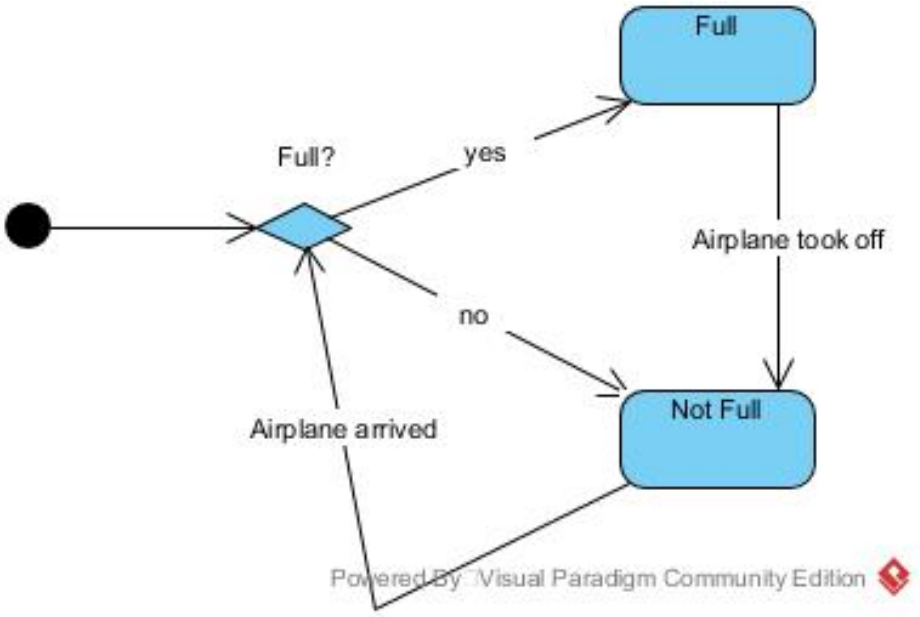

Fig. 7. Diagrama de estados de cada aeroporto

Ao inicializar, o estado do aeroporto é definido por se este está ou não cheio, se estiver cheio toma o estado de "cheio", se não toma o estado de "não cheio". Se estiver no estado "cheio" e um avião partir o estado do aeroporto muda para "não cheio". Caso o chegue um avião ao aeroporto e este tenha o estado de "não cheio", vai haver a verificação se depois da chegada este se encontra cheio ou não, caso se encontre cheio o estado e alterado para "cheio", se não o estado mantem-se.

4 Diagrama de Classes

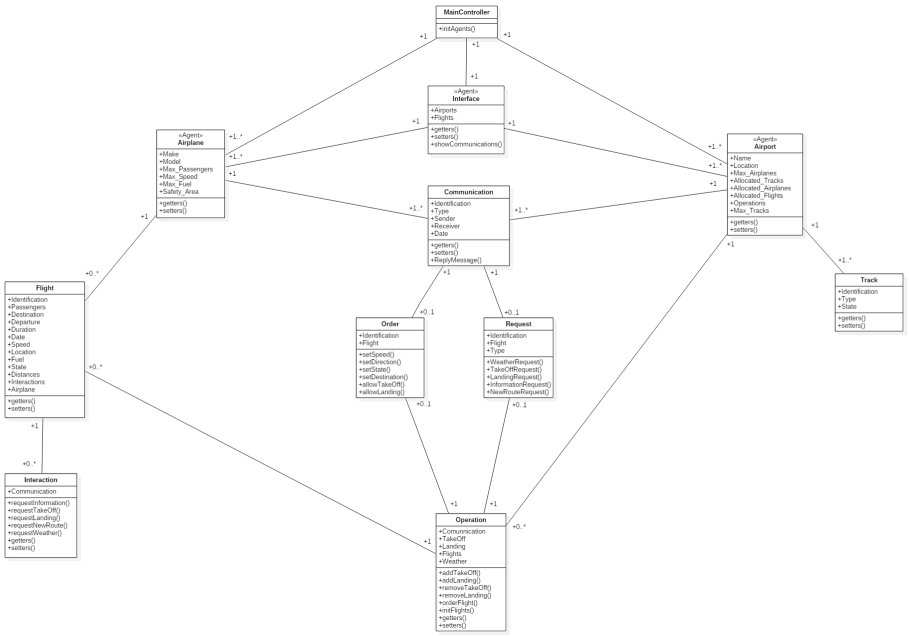

Fig. 8. Diagrama de Classes

Neste trabalho prático utilizamos o diagrama de classes para descrever a arquitetura do nosso sistema multi-agente. O objetivo é demonstrar como os vários agentes se relacionam e a que classes os mesmos estão afetados.

Este sistema está assente em três tipos de agentes base:

- Airplane
- Airport
- Interface

É responsabilidade da classe ***MainController*** a instanciação dos mesmos, definindo de forma aleatória o número de agentes que vai ser alocado em cada papel, sendo que cada papel é definido por uma classe que contem os atributos associados às características desse mesmo agente. Sempre que possível, atributos mais relevantes para o sistema vão ser gerados de forma aleatória, mantendo assim a fluidez que é esperada numa simulação. Atributos como o número de passageiros num voo, ou até mesmo o destino desse voo podem ser gerados de forma aleatória no agente ***Airplane***, o mesmo acontece com os atributos ***Max_Airplanes*** e ***Allocated_Tracks*** do agente ***Airport*** A simulação começa assim com um número prévio de aeroportos e de aviões que começam nesses mesmos aeroportos.

4.1 Agent Airplane

O agente ***Airplane*** representa cada uma das aeronaves existentes no sistema, e inicialmente vai ser alocado a um container com a capacidade de alojar mais agentes com o mesmo papel, e apenas um único agente com o papel de ***Airport***. Isto significa que cada container é um espaço de gestão de aeronaves, associadas a um único aeroporto.

Cada agente ***Airplane***, está dotado de um conjunto de características especificas, e que possibilitam a diferenciação entre os vários agentes com o mesmo papel. Essas características são referentes a atributos base de um modelo de aeronave, como a capacidade de passageiros, a velocidade máxima, a capacidade de combustível ou a área de segurança. A área de segurança de uma aeronave é um dos atributos mais relevantes deste agente, isto porque permite definir uma distância mínima entre outros agentes antes de proceder à tomada de uma decisão.

Cada agente ***Airplane*** estende as propriedades de uma classe voo, que é responsável por identificar uma viagem no sistema, e que foi criada no aeroporto de origem. A classe voo contem características como o local de partida e destino, localização atual do agente e sua velocidade, combustível atual da aeronave, número de passageiros, distâncias entre outros agentes, e comunicações realizadas com os mesmos.

A necessidade de guardar um histórico de comunicações entre os agentes, possibilita que essa informação seja enviada para o agente ***Interface*** e assim ser apresentada para o utilizador. Esse histórico é garantido pelo atributo **Interactions**, que vai ser uma lista de objetos da classe **Interaction**.

Os comportamentos que constituem o agente, possibilitam interações entre vários agentes em diferentes instantes de "tempo". O objetivo é garantir que um determinado voo chegue em segurança ao seu destino, evitando colisões com outros agentes, utilizando para tal métodos de negociação e comunicação. Cada agente tem assim a capacidade de fazer pedidos para agentes do mesmo papel, ou agentes ***Airport***. Esses pedidos podem ser:

- requestInformation: É feito um pedido de broadcast para todos os agentes ***Airplane***, com o objetivo de obter informações dos agentes que o rodeiam. Essa informação é essencial, no caso de ser necessário determinar a distância em relação a outros agentes de modo a evitar colisões.
- requestTakeOff: É feito um pedido de aterragem ao agente ***Airport***, com o objetivo de informar aquela entidade da necessidade de realizar uma aterragem. Esse pedido é concedido posteriormente por parte do agente ***Airport***, após uma análise de prioridades entres os diferentes voos.
- requestLanding: Tem um comportamento semelhante ao pedido de **requestTakeOff**.
- requestNewRoute: Na eminência de uma colisão entre diferentes voos, é feita uma negociação para que seja alterado algum tipo de parâmetro como a velocidade ou a direção, com o objetivo de evitar a colisão.

– requestWeather: O planeamento de uma rota durante o começo de um voo, implica um pedido de informação sobre o estado do tempo ao agente ***Airport***.

4.2 Agent Airport

Este agente faz a gestão das aeronaves e dos voos associados ao seu espaço aéreo. Possuem assim um conjunto de atributos que definem as caracteristicas fisicas desse aeroporto.

– Max_Airplanes: Define o número máximo de aeronaves em terra.
– Max_Tracks: Define o número máximo de pistas que um aeroporto possui
– Allocated_Tracks: Guarda uma lista com o estado atual de cada uma das pistas. Este atributo é uma lista de objetos da classe Track, e que permite fazer uma gestão de todas as pistas do aeroporto. Exemplo: Determinar se uma pista vai ser alocada para aterragens ou partidas.
– Allocated_Airplanes: Indica o número atual de aeronaves existentes no aeroporto.
– Allocated_Flighs: È um atributo lista, referente aos voos que estão associados ao aeroporto naquele momento. Este atributo pode ser obtido com base nos agentes ***Airplane*** existentes no container atual, e que estão associados a um determinado voo.
– Operations: É uma lista de objetos, com todas as operações realizadas até ao momento pelo aeroporto. Esta lista expande as propriedades da classe Operation, que é responsável pela implementação das ações realizadas pelo agente ***Airplane***. Em simultâneo permite manter um histórico de operações.

Classe Operation A classe Operation, indica que um determinado aeroporto pode realizar uma ou mais operações, ou seja, é uma classe que faz a gestão das ações no agente. ***Airport***. Esta classe é composta pelos atributos:

– Communication: Atributo que guarda a comunicação atual entre o agente ***Airport***, e os agentes ***Airplane***.
– TakeOff: É uma lista de pedidos para levantar voo. Esta lista é gerida e mantida pelo aeroporto, com base num conjunto de propriedades e prioridades.
– Landing: É uma lista de pedidos para aterragem de vários voos. Esta lista é gerida e mantida pelo aeroporto, com base num conjunto de propriedades e prioridades. Os pedidos de aterragem são feitos pelos agentes ***Airplane***, quando está no estado de voo.
– Flights: Lista de objetos da classe Flight, que guarda o estado atual de todos os voos.
– Weather: Atributo que guarda o estado do tempo, quando requisitado por um determinado agente ***Airplane***.

Esta classe é assim responsável pela gestão dos voos e pela sua inicialização. Devendo em qualquer altura responder a pedidos feitos por parte de outras aeronaves, ou se necessário dar ordens a essas mesmas aeronaves.

Os diferentes tipos de comunicação suportados pela classe Operation, implica que esta seja dotada de modificar ou criar instâncias de objetos das classes Request e Order, que têm como objetivo guardar as respostas relativamente ao tipo de pedido feito por parte de outros agentes.

4.3 Agent Interface

O agente ***Interface***, tem a responsabilidade de obter informações relativas aos diferentes agentes, e apresentar de forma intuitiva essa informação aos utilizadores.

A informação a apresentar deve ser relativa à comunicações entre os diferentes agentes, e características dos mesmos. Para que isto seja possível, este agente deve ser implementado numa arquitetura do sistema que o possibilite de obter informações relativamente aos restantes agentes, independentemente da localização dos mesmos, de entre os diferentes containers.

4.4 Class Communication

Classe base, usada como "pacote" da comunicação entre agentes ***Airplane*** e ***Airport***. Esta classe é composta por atributos necessários para constituir uma comunicação, como o remetente dessa comunicação, o destinatário que pode ser mais do que um agente em simultâneo, e o tipo de comunicação que vai ser realizado. O tipo de comunicação pode ser definido por pedidos ou ordens. Os pedidos são referentes à classe Request, e vão ser usados na sua totalidade pelo agente ***Airplane***, com o objetivo de fazer pedidos a outros agentes ***Airplane***, ou a agentes ***Airport***. As ordens vão ser criadas por agentes ***Airport***, e têm como objetivo definir comportamentos de voo nos agentes ***Airplane***.

Class Request Possibilita que os agentes ***Airplane*** realizem pedidos a agentes ***Airport***, ou a outros agentes com o mesmo papel. No caso em que o pedido é feito a um aeroporto, pode tratar-se de um pedido de informação temporal (**weatherRequest**), pedido de aterragem(**LandingRequest**) ou um pedido para levantar voo(**TakeOffRequest**). Caso o pedido seja para outras aeronaves, pode tratar-se de um pedido para obter informações relativamente aos restantes voos(**informationRequest**), ou um pedido para que outra aeronave mude a sua rota. Conforme referido anteriormente, um pedido de **informationRrequest**, implica algum tipo de negociação entre diferentes aeronaves, para que a nova rota seja definida. Este tipo de pedido tem como objetivo evitar as colisões entre diferentes agentes ***Airplane***.

Class Order O agente ***Airport***, tem a possibilidade de definir ordens, que influenciam o movimentação dos agentes ***Airplane***. Esta classe recebe parâmetros como a velocidade pretendida, direção, destino, ou estado de voo. Permite também definir quando é que um determinado voo pode aterrar ou levantar no aeroporto.

Estas duas classes são assim usadas como atributos da classe Communication, funcionando como modelos que possibilitam o armazenamento de informações, que estão relacionadas com o tipo de comunicação feita.