

# Agentes Inteligentes

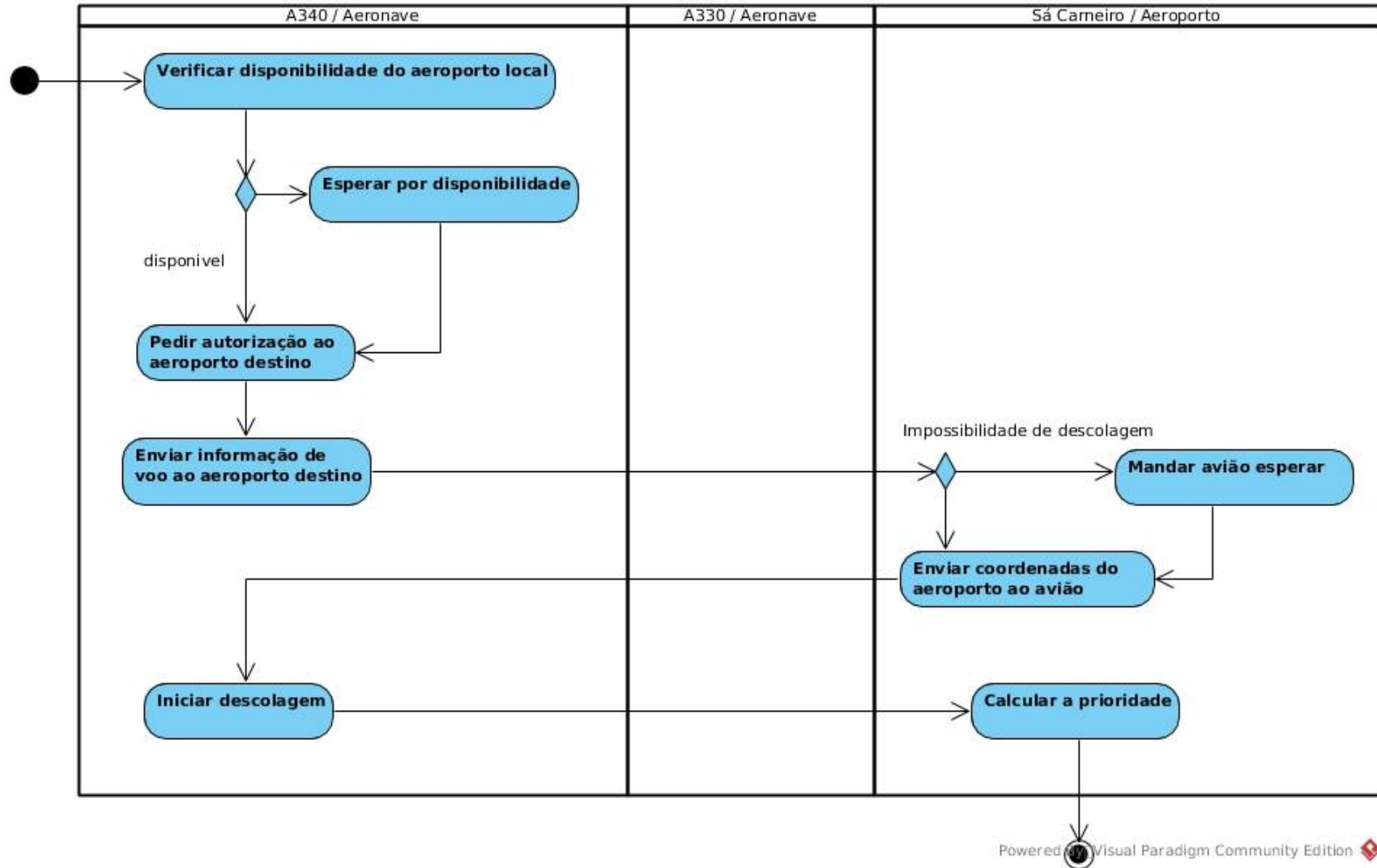
Gestão de tráfego aéreo

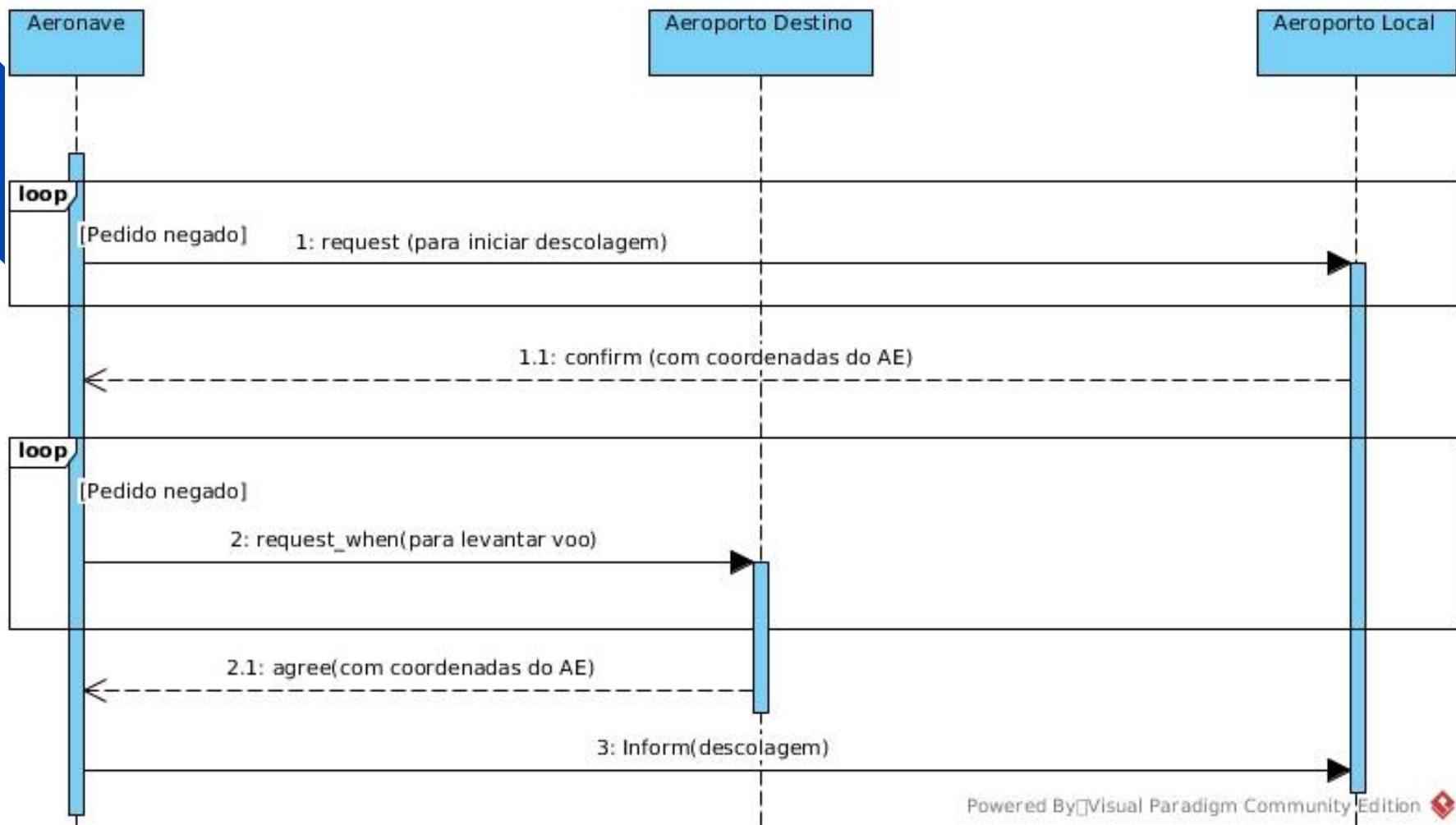
Carlos José Gomes Campos a74745  
José Pedro Ferreira Oliveira a78806  
Ludgero da Silva Diogo pg38417



# Principais estágios:

- Descolagem
- Em viagem
- Aterragem







# Agente Aeronave

O comportamento Descolagem é um TickerBehaviour responsável por enviar pedidos para autorização de início de viagem aos Agentes Estação:

```
if(velocidade == 0 && !autorizacaoPartida){  
    Envia REQUEST ao AE de origem;  
  
    }else if(velocidade == 0 && autorizacaoPartida && !autorizacaoChegada){  
  
        Envia REQUEST_WHEN com o nome + coordenadas + velocidade;  
  
    }else if(velocidade == 0 && autorizacaoPartida && autorizacaoChegada){  
  
        Envia INFORM ao AE de origem;  
  
    }
```



# Agente Estação

O comportamento receberPedidos é um CyclicBehaviour que processa todos os pedidos que chegam ao Agente Estação:

CASE : REQUEST

```
if(condMeteo && nrPistasOcupadas < totalPistas){
```

```
    Aloca pista para descolar;
```

```
    Desalocar estacionamento;
```

```
    Enviar CONFIRM;
```

```
}
```



CASE : REQUEST\_WHEN

```
if(condMeteo && nrEstOcupados < totalEstacionamentos){
```

```
    Aloca estacionamento;
```

```
    Calcula tempo de chegada;
```

```
    Atualiza lista de prioridades;
```

```
    Enviar AGREE;
```

```
}
```

CASE : INFORM

```
    Desaloca pista;
```

A340 / Aeronave

A330 / Aeronave

Sã Carneiro / Aeroporto

Enviar coordenadas em broadcast

Receber coordenadas de outros aviões

Verifica posição relativa a outros aviões

Está na AP

Verificar direção da rota do outro avião

Existe colisão frontal

Existe colisão de rotas

Virar direita

Ver quem está mais perto do destino

Diminuir velocidade

Está mais longe do destino

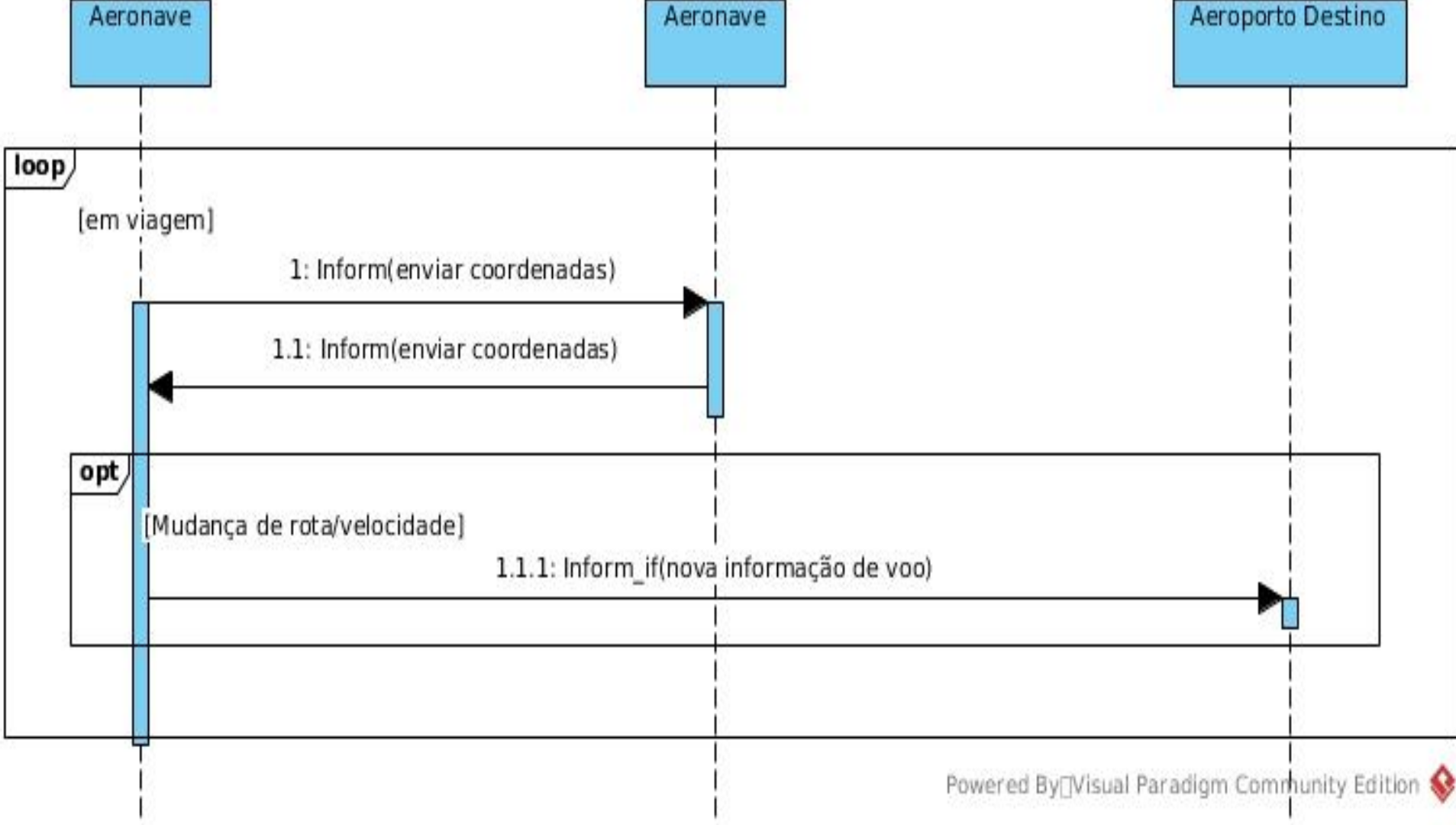
Informar aeroporto destino da alteração de rota/velocidade


Aumentar velocidade

Recalcular prioridade


Segue viagem



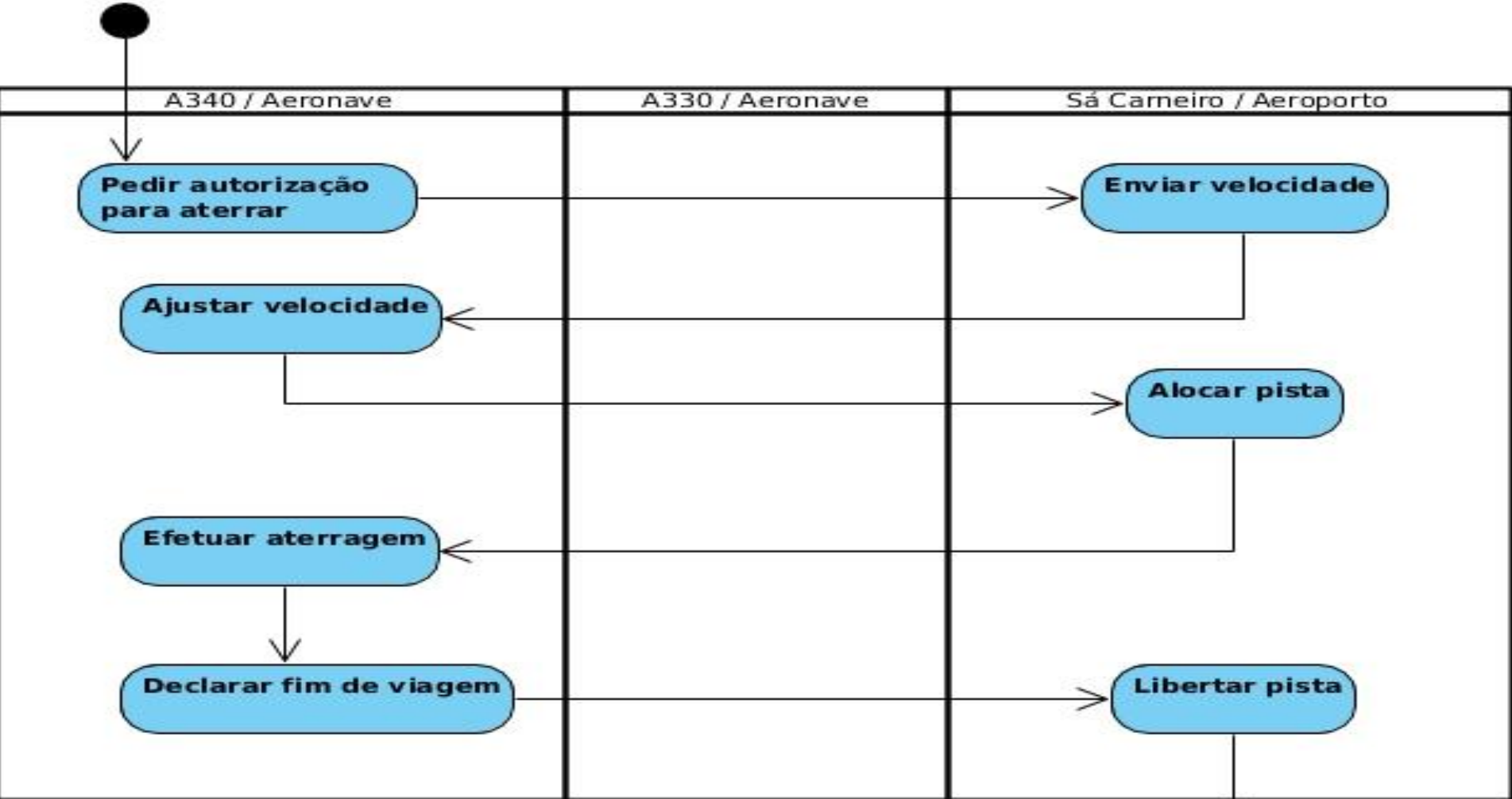




```
private class EnviarCoords extends TickerBehaviour {  
    ...  
    protected void onTick() {  
        ...  
        sd.setType("Plane");  
        ...  
        if(!provider.equals(this.myAgent.getAID())){  
            ACLMessage msg = new ACLMessage(ACLMessage.INFORM);  
            msg.addReceiver(provider);  
            msg.setContent(cinformaçãoViagem);  
            send(msg);  
        }  
        ....  
    }  
}
```



```
private class ReceberMsg extends CyclicBehaviour {
    public void action() {
        ...
        if (msg != null && msg.getPerformative() == ACLMessage.INFORM) {
            double dist = calcular distância entre aviões
            if(dist<=zonaAlerta) {
                double ang =calcular angulo entre retas
                if(choque de frente)
                    Curvar à direita 45°
                else if(há choque)
                    O mais longe acelera e o mais perto abranda.
            }
        }
    }
}
```



Aeronave

Aeroporto Destino

opt

[Está na AP do AE]


loop

[Autorização Aterragem]


1: Propose(pedido de aterragem)

1.1: Accept\_proposal(enviar velocidade)

2: Inform(fim de viagem)

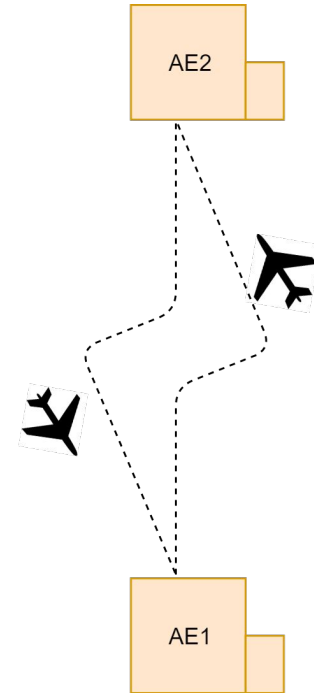
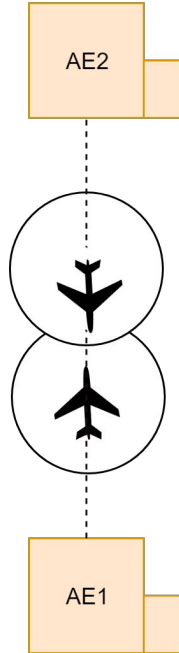


```
private class Aterragem extends CyclicBehaviour{
    public void action() {
        if(dentroAPdaAE){
            sd.setType(aes.get(conta+1));
            ...
            ACLMessage msg2 = new ACLMessage(ACLMessage.PROPOSE);
            ...
            msg2.setContent(name);
            send(msg2);
        }
        ...
        if(vimViagem){
            sd.setType(aes.get(conta+1));
            ...
            ACLMessage msg2 = new ACLMessage(ACLMessage.INFORM);
            ...
            msg2.setContent(name);
            send(msg2);
        }
        reset a variáveis...
    }
}
```



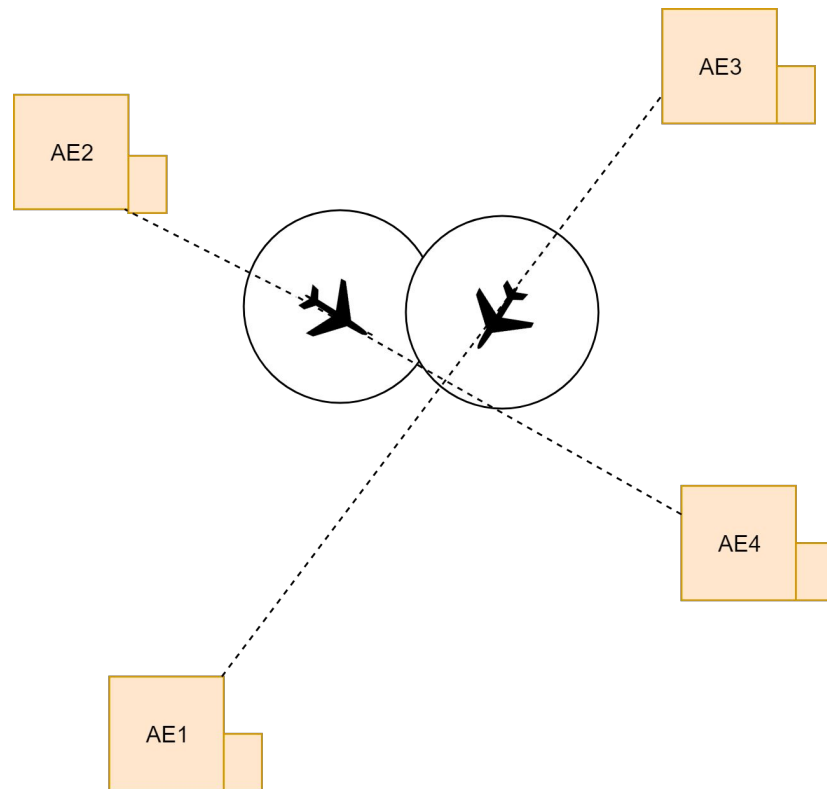
```
private class receberPedidos extends CyclicBehaviour {  
    ...  
    else if(msg != null && msg.getPerformative() == ACLMessage.PROPOSE &&  
nrPistasOcupadas<totalPistas) {  
        nrPistasOcupadas++;  
        resp.setPerformative(ACLMessage.ACCEPT_PROPOSAL);  
        double vel=getVelocidade(msg.getContent());  
        resp.setContent(vel+"");  
        send(resp);  
    }else if(msg != null && msg.getPerformative() == ACLMessage.INFORM) {  
        String n=msg.getContent();  
        removeLista(n);  
        nrPistasOcupadas--;  
    }  
    ...  
}
```

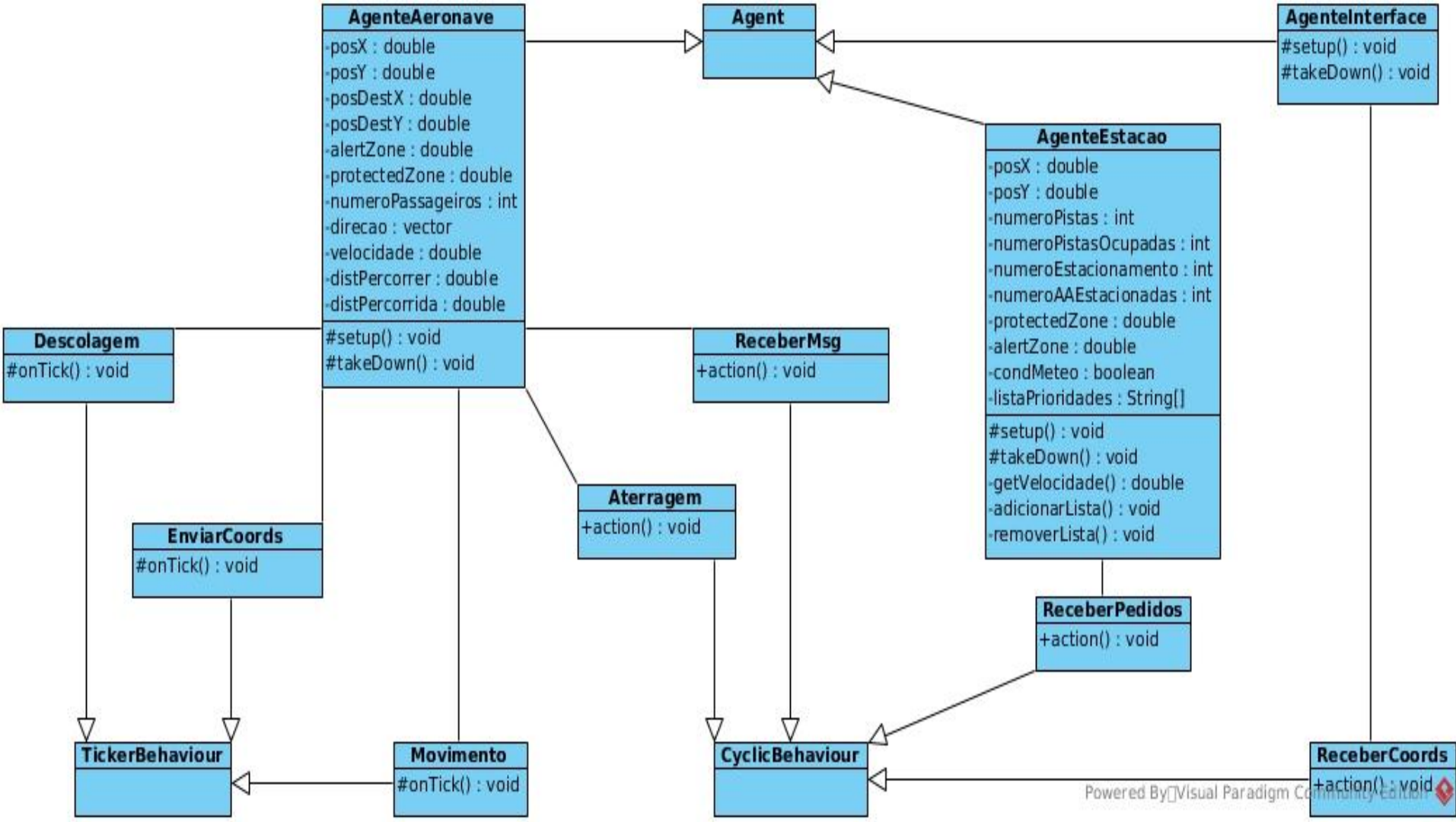
# Colisão frontal



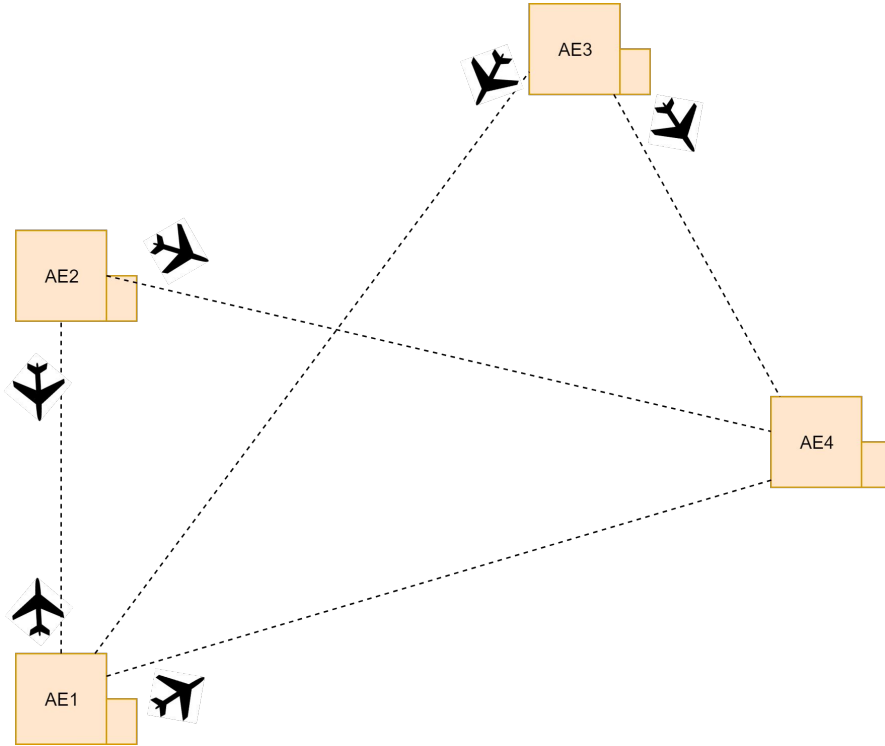


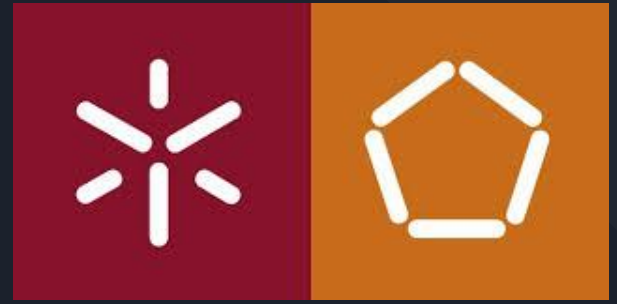
# Sobreposição de rotas





# Caso de teste





# Agentes Inteligentes

Gestão de tráfego aéreo

Carlos José Gomes Campos a74745  
José Pedro Ferreira Oliveira a78806  
Ludgero da Silva Diogo pg38417