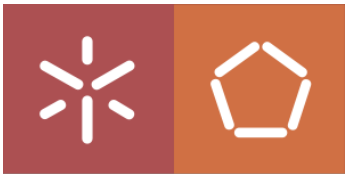


Universidade do Minho
Escola de Engenharia

Sistemas Inteligentes

Sistema Multiagente para Gestão de Transplantes de Órgãos

Clara Cunha

Leonor Amorim

Manuel Carvalho

Nuno Matos

Mestrado em Engenharia Biomédica - Informática Médica

3 de janeiro de 2025

Resumo

O presente trabalho propõe o desenvolvimento de um sistema multiagente para a gestão de transplantes de órgãos, implementado em *Python* com recurso à biblioteca SPADE. O sistema é composto por agentes que simulam diferentes entidades envolvidas no processo de transplante. Através de uma arquitetura modular, foram implementadas funcionalidades que incluem a alocação de órgãos com base em critérios de compatibilidade sanguínea, grau de urgência e logística de transporte, bem como mecanismos de contingência para lidar com eventuais falhas operacionais. A validação do sistema foi realizada mediante a simulação de diversos cenários representativos, os quais demonstraram a sua eficácia na coordenação das operações.

Palavras-chave: Sistemas multiagente, Transplantes de órgãos

Conteúdo

1 Introdução

O presente projeto consiste na elaboração de um sistema multiagente destinado à gestão de transplantes de órgãos, no qual agentes inteligentes são responsáveis por coordenar a alocação de órgãos entre dadores e recetores.

Este relatório inicia-se com a exposição do problema em análise, incluindo a identificação dos requisitos a cumprir. Segue-se a etapa de conceção da solução, onde são apresentadas a estratégia e a linha de raciocínio adotadas, bem como a arquitetura do sistema multiagente desenvolvido e a descrição do seu funcionamento. Adicionalmente, são descritos os testes realizados em diversos cenários com vista à validação do sistema. Por fim, são apresentadas algumas reflexões sobre o trabalho finalizado, destacando os principais desafios enfrentados e propondo sugestões e recomendações para a melhoria futura do sistema.

2 Análise e Especificação

2.1 Descrição do Problema

O projeto propõe a criação de um sistema multiagente para a gestão de transplantes de órgãos, no qual agentes inteligentes são responsáveis por coordenar a alocação de órgãos entre dadores e recetores. A solução deve considerar múltiplos fatores, tais como a compatibilidade entre dador e recetor, a urgência clínica dos pacientes e os requisitos logísticos necessários para garantir a viabilidade do transplante. Deste modo, o problema centra-se no desenvolvimento de agentes capazes de interagir e cooperar, culminando na criação de um sistema funcional que satisfaça todos os requisitos identificados.

2.2 Especificação dos Requisitos

Para atingir os objetivos propostos, o sistema multiagente deverá possuir as seguintes funcionalidades principais:

- Identificação do recetor mais adequado aquando da notificação da disponibilidade de um órgão, considerando a compatibilidade sanguínea, a urgência clínica e a proximidade geográfica entre o doador e o recetor.

- Verificação da disponibilidade de salas cirúrgicas e de equipas médicas no hospital do recetor selecionado, garantindo que todas as condições necessárias para a realização do transplante sejam cumpridas dentro do prazo exigido pela viabilidade do órgão.
- Implementação de mecanismos que permitem redirecionar o órgão para outro hospital, caso o hospital inicialmente designado não disponha dos recursos necessários no tempo requerido.
- Coordenação da logística de transporte para assegurar que o órgão chegue atempadamente ao hospital de destino, considerando a distância entre o local de origem do órgão e o hospital recetor.
- Simulação do tempo de transporte e do tempo de resposta do sistema para assegurar a viabilidade do órgão ao longo de todo o processo de transferência.
- Implementação de medidas de contingência em casos de falhas logísticas ou incompatibilidades, como a redistribuição do órgão para outro paciente elegível.

3 Arquitetura

A arquitetura do sistema é representada através de diagramas UML (*Unified Modeling Language*), os quais fornecem uma visão abrangente da estrutura e do comportamento dos seus componentes. Estes diagramas são fundamentais para compreender como os elementos interagem, como as funcionalidades são organizadas e como os processos fluem dentro do sistema.

A Figura 1 apresenta o diagrama de classes, que ilustra a estrutura estática do sistema. Este diagrama destaca as classes existentes, os respetivos atributos e métodos, bem como as relações entre elas, permitindo uma visão clara das responsabilidades de cada componente.

Por sua vez, o diagrama de atividades, ilustrado na Figura 2, descreve os fluxos de trabalho e os processos dinâmicos executados pelos agentes do sistema. Este diagrama detalha as diferentes etapas envolvidas e a forma como estas se interligam para alcançar os objetivos do sistema.

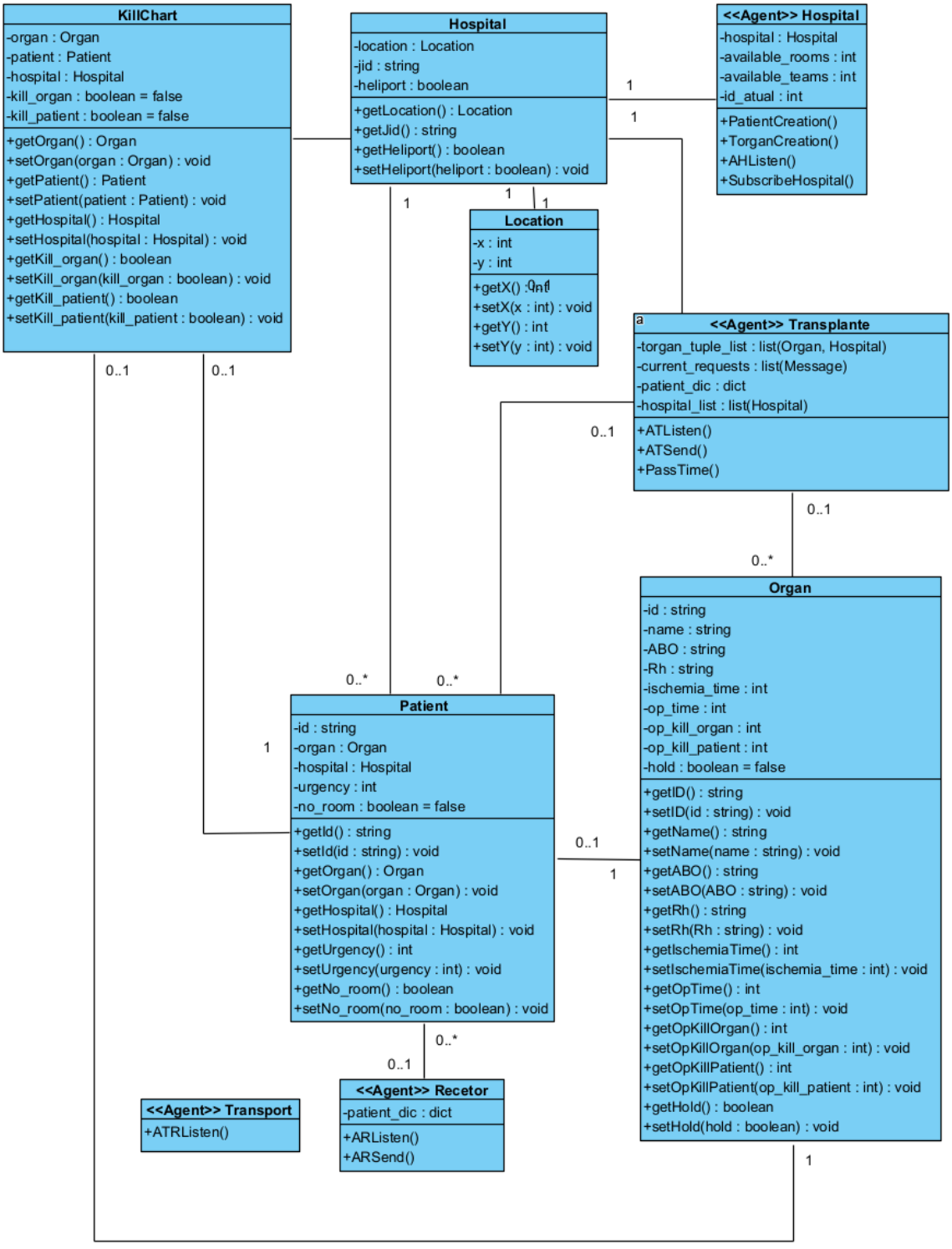


Figura 1: Diagrama de Classes.

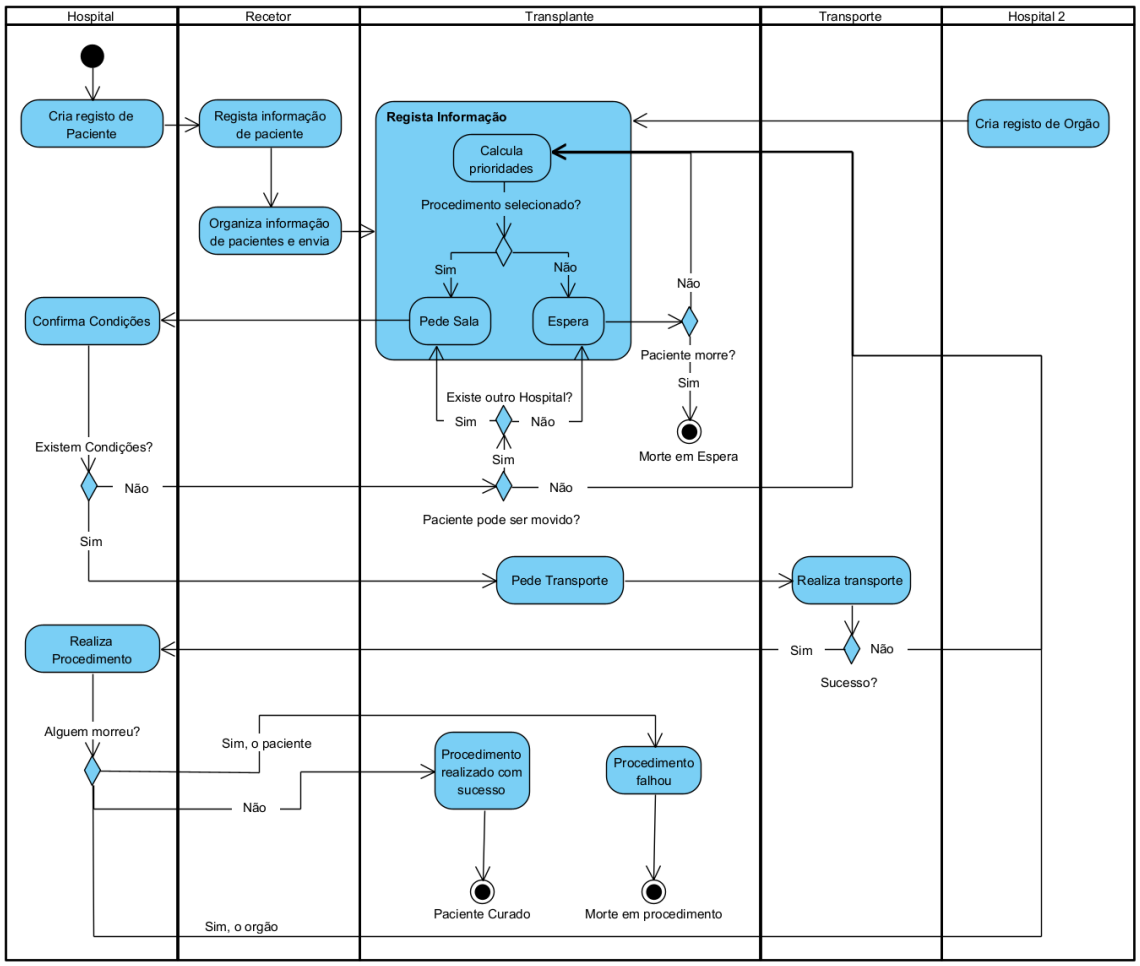


Figura 2: Diagrama de Atividades.

4 Implementação

O ambiente de trabalho utilizado para o desenvolvimento do sistema foi o *PyCharm*, utilizando a linguagem *Python* na versão 3.9. A biblioteca SPADE (*Smart Python Agent Development Environment*) foi empregada para criar agentes inteligentes e permitir a comunicação entre eles. Adicionalmente, o sistema operativo utilizado foi o *Windows 11*.

4.1 Parâmetros

Para permitir a adaptação e avaliação do sistema em diferentes ambientes, foi desenvolvido o ficheiro `parameters.py`, no qual se definem diversos parâmetros configuráveis do sistema, descritos a seguir.

Definição dos parâmetros associados ao órgão:

- **organ_type_dic**: Dicionário que especifica atributos essenciais para cada tipo de órgão, como tempo máximo de isquemia, duração do procedimento cirúrgico e probabilidade de complicações na viabilidade do órgão e do paciente.
- **ABO**: Lista com os diferentes tipos sanguíneos possíveis.
- **RH**: Lista com os diferentes fatores Rh possíveis.

Definição dos parâmetros associados ao transporte:

- **speed_hellichoppa**: Velocidade do helicóptero de transporte, em km/h.
- **speed_car**: Velocidade do carro de transporte, em km/h.
- **failure_rate**: Taxa de falha para os órgãos durante o transporte, em percentagem.
- **delay_rate**: Taxa de atraso nos transportes, em percentagem.

Definição dos parâmetros associados ao hospital:

- **hospital_number**: Número de hospitais no sistema.
- **min_rooms**: Número mínimo de salas de operação em cada hospital.
- **max_rooms**: Número máximo de salas de operação em cada hospital.
- **min_teams**: Número mínimo de equipas médicas disponíveis em cada hospital.
- **max_teams**: Número máximo de equipas médicas disponíveis em cada hospital.
- **min_pat_period**: Período mínimo entre criação de novos pacientes.
- **max_pat_period**: Período máximo entre criação de novos pacientes.

Definição dos parâmetros associados ao paciente:

- **min_start_urg**: Nível mínimo de urgência do paciente na sua criação.
- **max_start_urg**: Nível máximo de urgência do paciente na sua criação.
- **urgency_increase_chance**: Probabilidade de aumento da urgência a cada hora, em percentagem.

- **urg_heli**: Percentagem de urgência acima da qual o transporte por helicóptero é considerado.
- **urg_thresh**: Percentagem mínima de urgência para que o paciente possa ser transferido para outro hospital.

Este ficheiro inclui ainda os parâmetros `XMPP_SERVER` e `PASSWORD`, permitindo que o sistema seja facilmente portável para diferentes ambientes de execução.

4.2 Classes

As classes implementadas no sistema dispõem de métodos `getters` e `setters`, que permitem o acesso e a atualização controlada dos atributos, e o método `str` para oferecer uma representação textual customizada de cada objeto. Seguem-se as descrições das classes implementadas.

4.2.1 Órgão

A classe `Organ` representa um órgão com atributos relacionados à sua identificação e características operacionais:

- **id**: Identificador único do órgão.
- **name**: Nome do tipo de órgão (e.g., rim, coração).
- **ABO**: Tipo sanguíneo do órgão.
- **Rh**: Fator Rh do órgão.
- **ischemia_time**: Período em que o órgão pode permanecer viável fora do corpo.
- **op_time**: Tempo necessário para a operação de transplante.
- **op_kill_organ**: Probabilidades de falha do órgão durante a operação.
- **op_kill_patient**: Probabilidade de morte do paciente devido à operação.
- **hold**: Estado do órgão, indicando se este se encontra em espera ou em uso.

Ao ser instanciado, um objeto desta classe inicializa automaticamente os seus valores com base em escolhas aleatórias e nas informações do dicionário `organ_type_dic` e nas listas ABO e RH provenientes do ficheiro de parâmetros.

4.2.2 Localização

A classe `Location` representa uma localização definida por duas coordenadas cartesianas, que são geradas aleatoriamente ao instanciar a classe, com valores entre 1 e 100:

- **x**: Coordenada X da localização.
- **y**: Coordenada Y da localização.

4.2.3 Hospital

A classe `Hospital` representa um hospital no sistema e contém as seguintes informações:

- **location**: Instância da classe `Location`, que armazena informações sobre a localização do hospital.
- **jid**: Identificador único do hospital, representado por uma *string*.
- **heliport**: Indicador booleano que determina se o hospital possui heliporto, com base na probabilidade definida pelo parâmetro `heliport_chance` no ficheiro de parâmetros.

4.2.4 Paciente

A classe `Patient` representa um paciente que necessita de um transplante de órgão, armazenando as seguintes informações:

- **id**: Identificador único do paciente.
- **organ**: Órgão que o paciente necessita, representado por uma instância da classe `Organ`.
- **hospital**: Hospital onde está localizado paciente, representado por uma instância da classe `Hospital`.
- **urgency**: Nível de urgência do caso do paciente, gerado aleatoriamente entre os valores definidos pelas constantes `min_start_urg` e `max_start_urg` no ficheiro de parâmetros e usada para priorizar o atendimento do paciente.
- **no_room**: Um valor booleano que indica se o paciente não tem quarto disponível no hospital, sendo inicialmente definido como *False*.

4.2.5 Morte

A classe `KillChart` regista os riscos associados a um transplante de órgão, tanto para o órgão quanto para o paciente. Esta classe é necessária para documentar os resultados dos transplantes, permitindo monitorizar o impacto das operações em cada um dos elementos envolvidos, e armazena as seguintes informações:

- **organ**: Órgão a ser transplantado.
- **patient**: Paciente que recebe o transplante.
- **hospital**: Hospital responsável pelo procedimento de transplante.
- **kill_organ**: Booleano que indica se o transplante resulta na destruição do órgão, inicializado como *False*.
- **kill_patient**: Booleano que indica se o transplante causa a morte do paciente, inicializado como *False*.

4.3 Funções

As funções implementadas têm o objetivo de otimizar o processo de transplantes de órgãos, ajudando a calcular tempos de transporte entre hospitais, verificar a compatibilidade sanguínea entre órgãos e pacientes, além de determinar a prioridade na alocação de órgãos com base em fatores como urgência e localização. A seguir, são descritas as principais funções desenvolvidas.

4.3.1 Tempo de transporte

A função `calc_time` calcula o tempo de viagem entre dois hospitais, considerando a distância entre eles, o meio de transporte e a urgência do paciente. Se ambos os hospitais possuem heliporto e o caso for urgente, o transporte será feito de helicóptero; caso contrário, será feito via terrestre.

A função `three_way_time` calcula o tempo de transporte máximo entre dois hospitais e um destino comum, utilizando a função `calc_time`, para transportar um órgão e um paciente, levando em consideração a urgência do caso. Esta retorna o maior tempo de viagem calculado, pois esse será o tempo total de deslocação.

4.3.2 Compatibilidade sanguínea

A função `blood_compatibility` verifica se o órgão e o paciente são compatíveis em termos de grupo sanguíneo e fator Rh, retornando *True* se o órgão for compatível com o paciente e *False* caso contrário.

4.3.3 Prioridade

A função `priority_score` calcula a pontuação de prioridade para um órgão com base na urgência do paciente, no tempo de transporte (utilizando a função `calc_time`) e no tempo de isquemia do órgão. Quanto menor o valor retornado, maior a prioridade do paciente para receber o órgão.

A função `priority_neworgan` seleciona o paciente com maior prioridade para receber um órgão. Esta filtra os pacientes que não possuem um órgão atribuído e que têm compatibilidade sanguínea com o órgão, utilizando a função `blood_compatibility`. Para cada paciente compatível, calcula-se a pontuação de prioridade usando a função `priority_score`. A lista de pacientes é então ordenada pela pontuação de prioridade e o paciente com a maior pontuação é selecionado. Assim, a função retorna um tuplo com o órgão, o hospital e o paciente selecionados.

A função `priority_newpatients` calcula a prioridade de alocação de órgãos para múltiplos pacientes, retornando uma lista de pacientes com maior prioridade para cada órgão disponível. Para cada tuplo órgão-hospital na lista de tuplos, a função verifica se o órgão não está em espera e calcula a prioridade de alocação para os pacientes do hospital correspondente usando a função `priority_neworgan`. O órgão é atribuído ao paciente mais prioritário, se encontrado, e o paciente é marcado como "em espera". A função retorna uma lista de tuplos com o órgão, o hospital e o paciente com maior prioridade para receber o órgão.

4.3.4 Melhor hospital

A função `best_hospital_list` ordena os hospitais por tempo de transporte entre o hospital do paciente, o hospital do órgão e o destino final, calculado usando a função `three_way_time`. O objetivo é encontrar o hospital mais adequado para realizar o transplante, considerando a urgência do caso e o tempo total de transporte entre os hospitais envolvidos.

4.4 Agentes

4.4.1 Agente Transplante (AT)

O Agente Transplante (AT) é responsável por coordenar a gestão de órgãos disponíveis para transplante, recebendo notificações de novos órgãos e verificando a compatibilidade com os pacientes na lista de espera. Ele mantém as seguintes listas:

- **torgan_tuple_list**: Lista de tuplos que armazena informações sobre órgãos disponíveis e os hospitais onde se encontram.
- **current_requests**: Lista que armazena mensagens de solicitação de salas para transplante, pendentes de resposta.
- **patient_dic**: Dicionário que organiza os pacientes por tipo de órgão requerido.
- **hospital_list**: Lista de hospitais no sistema.

O agente opera com três comportamentos principais:

- **ATListen(CyclicBehaviour)**: Este comportamento cíclico aguarda mensagens e processa-as com base na *performative* associada:
  - **Inscrição de hospitais (*subscribe*)**: Recebe e processa inscrições de hospitais, adicionando-os à lista `hospital_list`.
  - **Processamento de novos órgãos (*subscribe_torgan*)**: Regista a chegada de novos órgãos ao sistema, adicionando o órgão e o hospital associado à lista `torgan_tuple_list`. A função `priority_neworgan` é utilizada para determinar o paciente mais compatível com o órgão. Caso um paciente compatível seja identificado, o órgão e o paciente são colocados em estado "hold" para evitar que sejam re-associados enquanto o processo de alocação está em andamento. Um pedido é então enviado ao hospital do paciente para verificar a disponibilidade de condições para o transplante, e este pedido é adicionado à lista `current_requests`.
  - **Atualização de lista de pacientes (*inform_patient_list*)**: Recebe uma lista de novos pacientes e integra-a no dicionário `patient_dic`. A função `priority_newpatients` é usada para determinar pacientes prioritários para os órgãos disponíveis. Caso um paciente compatível seja identificado, o órgão e o paciente são colocados em estado "hold" para evitar que sejam re-associados, e é enviado um pedido ao hospital do paciente para verificar a disponibilidade de condições. Este pedido é adicionado à lista `current_requests`.

– **Confirmação de condições (*confirm_room*)**: Após a confirmação de condições por parte do hospital, o AT solicita ao ATR a logística para entrega do órgão.

– **Recusa de condições (*refuse_room*)**: Caso o hospital recuse as condições para o transplante, o AT implementa medidas de contingência. Ele verifica a urgência do paciente, que é comparada com um limite (`urg_thresh`) (quanto maior o valor do nível de urgência, menos crítico é o paciente):

  1. Se o paciente não está numa situação crítica (o nível de urgência é maior que o limite de urgência), pode ser transportado para outro hospital. O AT procura outras salas disponíveis na `hospital_list`, excluindo o hospital original onde o paciente estava. A função `best_hospital_list` avalia os melhores hospitais disponíveis para o paciente. O melhor hospital da lista é escolhido e o AT solicita uma nova sala e equipa médica para o paciente. Este pedido é adicionado à lista `current_requests`.

  2. Se o paciente está em estado crítico, não pode ser transportado para outro hospital. Neste caso, o AT tenta encontrar outro paciente para o órgão através da função `priority_neworgan`. O paciente original é removido da lista de espera e marcando como "sem sala". Se um novo paciente for encontrado, o órgão e o novo paciente são colocados em estado de "hold" e é enviada uma mensagem ao hospital do paciente para verificar a disponibilidade de condições. Este pedido é adicionado à lista `current_requests`.

– **Confirmação de condições noutro hospital (*confirm_another_room*)**: Após a confirmação de que um paciente e um órgão podem ser alocados noutro hospital, o AT solicita ao ATR a logística para o transporte do órgão e do paciente.

– **Recusa de condições noutro hospital (*refuse_another_room*)**: Se não for possível encontrar um hospital com condições adequadas após tentar hospitais alternativos, o AT liberta o "hold" em órgãos e pacientes, e regista que não há hospitais adequados disponíveis.

– **Sucesso na operação (*confirm_op*)**: O órgão e o paciente são removidos da lista de espera.

– **Falha na operação (*failure_op*)**: A razão pode ser a morte do paciente, a destruição do órgão, ou ambos.

  1. Caso a falha seja por morte do paciente, o paciente é removido da lista e o órgão é libertado.

2. Caso a falha seja por destruição do órgão, o órgão é removido da lista e o paciente é libertado.

De seguida, o AT recalcula as prioridades dos pacientes e, caso um paciente compatível seja identificado, o órgão e o paciente são colocados em estado "hold" e é enviado um pedido ao hospital do paciente para verificar a disponibilidade de condições. Este pedido é adicionado à lista `current_requests`.

– **Falha no transporte (*failure_transport*)**: O órgão correspondente é removido da lista de órgãos e o paciente é tirado do "hold".

- **ATSend(CyclicBehaviour)**: Comportamento que envia as mensagens de solicitação de salas (armazenadas na lista `current_requests`) para os hospitais. Após o envio, a mensagem é removida da lista.

- **PassTime(PeriodicBehaviour)**: Comportamento que simula a passagem do tempo. Ele reduz o tempo de isquemia dos órgãos e aumenta a urgência dos pacientes. Se o tempo de isquemia de um órgão ou a urgência de um paciente chegar a zero, o órgão ou paciente é removido da lista.

4.4.2 Agente Recetor (AR)

O Agente Recetor (AR) é responsável por gerir informações sobre pacientes e coordenar a comunicação com o AT para atualizar a lista de pacientes que aguardam órgãos para transplante. Ele mantém um dicionário principal:

- **patient_dic**: Armazena listas de pacientes compatíveis para cada tipo de órgão.

O agente opera com dois comportamentos:

- **ARListen(CyclicBehaviour)**: Monitoriza continuamente mensagens recebidas, garantindo que o AR esteja sempre atualizado com as informações sobre novos pacientes e que elas sejam armazenadas corretamente para serem usadas posteriormente. São aguardadas mensagens do tipo *inform*, que contêm informações de um novo paciente associado a um órgão específico. O paciente é então adicionado à lista de pacientes do dicionário `patient_dic` e é inserido na chave correspondente ao tipo de órgão que ele necessita. Caso a mensagem recebida não seja do tipo esperado, o agente exibe um erro.

- **ARSend(PeriodicBehaviour)**: É executado de forma periódica, com um intervalo de 5 segundos. O AR prepara e envia uma mensagem para o AT com a lista atualizada de pacientes. Depois de enviar a mensagem, a lista de pacientes é limpa para garantir que o AR envie apenas dados atualizados.

4.4.3 Agente Transporte (ATR)

O Agente Transporte (ATR) é responsável por processar pedidos de transporte de órgãos e pacientes, avaliando as condições de transporte, calculando distâncias e tempos, e enviando mensagens para notificar o sucesso ou a falha das operações.

O agente mantém um comportamento:

- **ATRListen(CyclicBehaviour)**: Este comportamento executa uma ação cíclica, aguardando mensagens e processando-as com base na *performative* associada:

  - **Requisição de transporte de órgão (*request_transport*)**: Verifica a urgência do paciente e calcula o tempo necessário para o transporte através da função `calc_time`, considerando a distância entre os hospitais e o meio de transporte disponível (carro ou helicóptero). Se o tempo de transporte é calculado como zero, isso significa que o órgão já está no hospital do paciente, e o ATR notifica o sucesso do transporte sem a necessidade de deslocamento. Caso contrário, o ATR simula o transporte, considerando possíveis falhas e atrasos.

  1. Se ocorrer uma falha no transporte, o ATR simula o tempo de falha e notifica o AT sobre o problema para que este possa tomar medidas de contingência.

  2. Se não houver falha, o ATR simula o transporte com um possível atraso, e então notifica o hospital de destino sobre a conclusão bem-sucedida do transporte.

  - **Requisição de transporte de paciente (*request_patient_transport*)**: O ATR também lida com solicitações de transporte de pacientes, onde o transporte tanto do paciente quanto do órgão deve ser coordenado. O agente simula o tempo de transporte para ambos, considerando as condições de urgência e a infraestrutura dos hospitais envolvidos, e simula a viagem com possíveis atrasos.

4.4.4 Agente Hospital (AH)

O Agente Hospital (AH) é responsável pela gestão dos recursos de um hospital, incluindo a criação e a gestão de pacientes e órgãos, bem como a disponibilidade de salas cirúrgicas e equipas médicas para o procedimento operatório. O AH mantém as seguintes variáveis:

- **hospital**: instância da classe `Hospital`.
- **available_rooms**: número de salas cirúrgicas disponíveis para a cirurgia.
- **available_teams**: número de equipas médicas disponíveis para a cirurgia.
- **id_atual_pat**: identificador incremental para pacientes do hospital.
- **id_atual_organ**: identificador incremental para órgãos disponíveis no hospital.

O agente opera com vários comportamentos:

- **PatientCreation(PeriodicBehaviour)**: Cria pacientes periodicamente com um identificador único, que é composto pelo ID do hospital e um número sequencial de paciente. A cada execução, ele envia uma mensagem para o AR, informando a criação do paciente.
- **TorganCreation(PeriodicBehaviour)**: Cria órgãos periodicamente com um identificador único, que também é composto pelo ID do hospital e um número sequencial de órgão. Após a criação do órgão, o comportamento envia uma mensagem para o AT para notificar a criação do órgão.
- **SubscribeHospital(OneShotBehaviour)**: Inscreve-se, no AT, enviando informações sobre o hospital.
- **AHListen(CyclicBehaviour)**: Este comportamento cíclico é responsável por escutar as mensagens recebidas e processá-las com base na *performative* associada. As ações realizadas para cada tipo de mensagem são as seguintes:

  - **Requisição de sala (*request_room*) e *request_another_room***: Verifica se há salas e equipas disponíveis no hospital para realizar o procedimento. Caso haja disponibilidade, ele aloca os recursos necessários, diminuindo o número de salas e de equipas disponíveis, e envia uma resposta de confirmação. Caso contrário, envia uma resposta de recusa.

  - **Confirmação de transporte (*confirm_transport*)**: Simula o tempo da operação (com base no tempo de operação do órgão) e depois faz a atualização dos recursos do hospital, aumentando o número de salas e equipas disponíveis. O comportamento também simula possíveis falhas durante o transplante, como a morte do paciente ou do órgão. Por fim, envia mensagens de confirmação ao AT se o procedimento for bem-sucedido ou mensagens de falha caso haja problemas.

4.5 Execução Principal

A execução principal do código configura e inicializa os agentes do sistema de transplante, criando os agentes necessários para simular o processo de alocação de órgãos e transplantes. Inicia-se com a criação dos agentes principais (AT, AR, ATR) e a inicialização dos agentes hospitalares (AH), com base no número de hospitais definidos. Cada agente é iniciado de maneira assíncrona, e o código aguarda até que todos estejam registados no servidor XMPP. O sistema permanece em execução enquanto os agentes principais estiverem ativos. No final, a função `quit_spade()` é chamada para encerrar o ambiente de execução e desconectar do servidor XMPP.

5 Resultados e Avaliação

Nesta secção são apresentados os resultados obtidos a partir dos testes realizados no sistema, com o objetivo de avaliar a sua eficácia em diferentes cenários, considerando tanto o sucesso quanto as falhas potenciais que podem ocorrer durante o processo de alocação, transporte e transplante de órgãos.

5.1 Realização de um transplante com sucesso

Neste teste é verificado o processo completo de realização de um transplante de fígado, em que o hospital tem todos os recursos necessários e o procedimento ocorre sem falhas.

1. O hospital responsável pela operação começa o processo de confirmação das condições necessárias para realizar o procedimento.

```
Hospital 8 at Location [X=42, Y=81]:
Confirming conditions for Patient H8-P1: organ: liver; type: O+; urgency: 9.
Available rooms: 4; Available teams: 7.
```

2. O transporte do órgão é iniciado.

```
Confirmation from Hospital 4 at Location [X=94, Y=23] for Patient H8-P1: organ: liver; type: O+; urgency: 9.
Initiating transport for Organ H4-01: liver; type: AB+...
```

3. O órgão chega ao hospital onde o transplante será realizado.

```
Organ H4-01: liver; type: AB+ successfully arrived at Hospital 8 at Location [X=42, Y=81] for Patient H8-P1: organ: liver; type: O+; urgency: 9 by car in 0.39 hours, including a delay of 0.0 hours.
```

4. O órgão é transplantado com sucesso no paciente.

```
Patient H8-P1: organ: liver; type: O+; urgency: 9 was successfully implanted with Organ H4-01: liver; type: AB+.
```

```
Patient H8-P1: organ: liver; type: O+; urgency: 9 procedure successful in Hospital 8 at Location [X=42, Y=81]!
```

Exemplos de transplante para outros órgãos:

- Pulmão

```
Patient H15-P1: organ: lung; type: AB+; urgency: 9 was successfully implanted with Organ H15-02: lung; type: B+.
```

- Pâncreas

```
Patient H10-P2: organ: pancreas; type: AB+; urgency: 9 was successfully implanted with Organ H9-03: pancreas; type: AB+.
```

- Córnea

```
Patient H13-P2: organ: cornea; type: A+; urgency: 9 was successfully implanted with Organ H7-01: cornea; type: A+.
```

5.2 Hospital sem recursos para realização do transplante

5.2.1 Paciente crítico

Neste teste, o hospital inicialmente recusa as condições para realizar a cirurgia de transplante e, como o paciente é crítico, o órgão selecionado é redistribuído a outro paciente elegível.

1. O hospital recusa ter as condições necessárias para a realização da operação. O órgão é redirecionado para outro paciente.

```
Conditions not met for Patient H7-P1: organ: pancreas; type: AB+; urgency: 8 in Hospital 7 at Location [X=96, Y=90].
Searching for new patient for Organ H7-01: pancreas; type: B+...
```

2. O hospital do novo rector confirma ter as condições necessárias para a realização da operação e o transporte do órgão é iniciado.

```
Confirmation from Hospital 9 at Location [X=57, Y=79] for Patient H9-P1: organ: pancreas; type: B+; urgency: 9.
Initiating transport for Organ H7-01: pancreas; type: B+...
```

3. O órgão chega ao hospital onde o transplante será realizado.

```
Organ H7-01: pancreas; type: B+ successfully arrived at Hospital 9 at Location [X=57, Y=79] for Patient H9-P1: organ: pancreas; type: B+; urgency: 9 by car in 0.7 hours, including
 a delay of 0.19 hours.
```

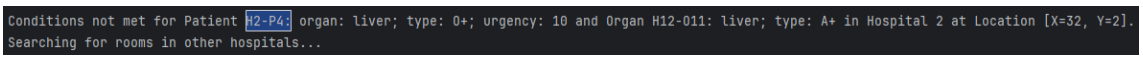
4. O órgão é transplantado com sucesso no paciente.

```
Patient H9-P1: organ: pancreas; type: B+; urgency: 9 was successfully implanted with Organ H7-01: pancreas; type: B+.
```

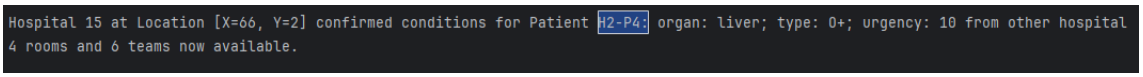
5.2.2 Paciente não crítico

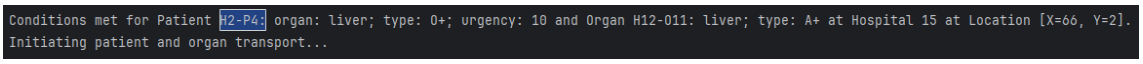
Neste teste, o cenário envolve um hospital que inicialmente recusa as condições para realizar a cirurgia de transplante. Como resultado, o paciente é transportado para outro hospital que possui as condições necessárias para a operação.

1. O hospital recusa ter as condições necessárias para a realização da operação. O sistema procura por outros hospitais para o paciente ser transplantado.

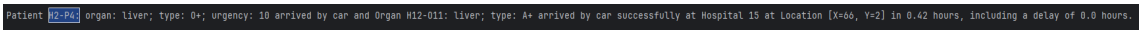


2. O hospital alternativo confirma ter as condições necessárias para a realização da operação e o transporte do órgão e do paciente é iniciado.

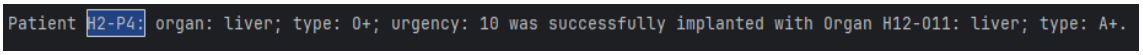




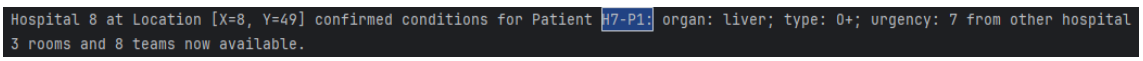
3. O órgão e o paciente chegam ao hospital onde o transplante será realizado.

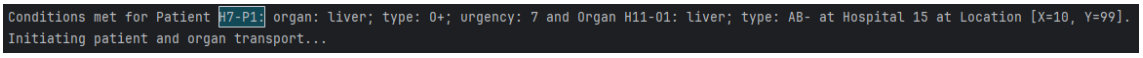


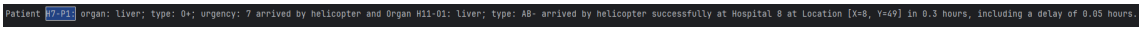
4. O órgão é transplantado com sucesso no paciente.

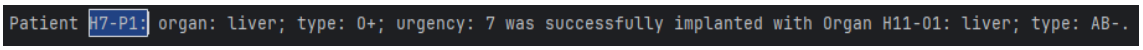


Neste teste o percurso é o mesmo, mas neste caso tanto o órgão quanto o paciente são transportados de helicóptero. Para facilitar a identificação dos helicópteros, os parâmetros `urg_heli` e `heliport_chance` no ficheiro de parâmetros foram aumentados.









5.3 Falha no transplante

5.3.1 Morte do paciente

Neste cenário é verificada a morte do paciente durante o transplante e a medida de contingência implementada.

1. O órgão chega ao sistema.

```
Organ H9-02: liver; type: B- info received
```

2. O hospital confirma ter as condições necessárias para a realização da operação e o transporte do órgão é iniciado.

```
Confirmation from Hospital 9 at Location [X=99, Y=66] for Patient H14-P3: organ: liver; type: AB-; urgency: 4.
Initiating transport for Organ H9-02: liver; type: B-...
```

3. O órgão chega ao hospital (com atraso) onde o transplante será realizado.

```
Organ H9-02: liver; type: B- successfully arrived at Hospital 14 at Location [X=59, Y=90] for Patient H14-P3: organ: liver; type: AB-; urgency: 4 by car in 0.35 hours, including a delay of 0.12 hours.
```

4. A cirurgia falha por morte do paciente.

```
Procedure for Patient H14-P3: organ: liver; type: AB-; urgency: 4 and Organ H9-02: liver; type: B- failure:
Patient died.
```

5. O órgão volta a estar disponível no sistema, é redistribuído a outro paciente e é iniciado o seu transporte.

```
Confirmation from Hospital 14 at Location [X=59, Y=90] for Patient H5-P8: organ: liver; type: O+; urgency: 7.
Initiating transport for Organ H9-02: liver; type: B-...
```

6. O órgão chega ao hospital onde o transplante será realizado com um pequeno atraso.

```
Organ H9-02: liver; type: B- successfully arrived at Hospital 5 at Location [X=56, Y=88] for Patient H5-P8: organ: liver; type: O+; urgency: 7 by car in 0.03 hours, including a delay of 0.01 hours.
```

7. O órgão é transplantado com sucesso no novo paciente.

```
Patient H5-P8: organ: liver; type: O+; urgency: 7 was successfully implanted with Organ H9-02: liver; type: B-.
```

5.3.2 Morte do órgão

Neste cenário é verificada a destruição do órgão durante o transplante e a medida de contingência implementada.

1. O hospital confirma ter as condições necessárias para a realização da operação e o transporte do órgão é iniciado.

```
Hospital 15 at Location [X=66, Y=2]:
Confirming conditions for Patient H15-P1: organ: heart; type: A+; urgency: 2.
Available rooms: 4; Available teams: 6.
```

```
Confirmation from Hospital 6 at Location [X=96, Y=86] for Patient H15-P1: organ: heart; type: A+; urgency: 2.
Initiating transport for Organ H6-02: heart; type: O+...
```

2. O órgão chega ao hospital onde o transplante será realizado.

```
Organ H6-O2: heart; type: O+ successfully arrived at Hospital 15 at Location [X=66, Y=2] for Patient H15-P1: organ: heart; type: A+; urgency: 2 by car in 0.45 hours, including a delay of 0.0 hours.
```

3. A cirurgia falha por destruição do órgão.

```
Procedure for Patient H15-P1: organ: heart; type: A+; urgency: 2 and Organ H6-O2: heart; type: O+ failure: Organ destroyed.
```

4. O paciente volta a estar disponível no sistema e é encontrado um novo órgão para ele.

```
Hospital 15 at Location [X=66, Y=2]:
Confirming conditions for Patient H15-P1: organ: heart; type: A+; urgency: 2.
Available rooms: 4; Available teams: 6.
```

5. O hospital confirma ter as condições necessárias para a realização da operação e o transporte do órgão é iniciado.

```
Confirmation from Hospital 11 at Location [X=15, Y=21] for Patient H15-P1: organ: heart; type: A+; urgency: 2.
Initiating transport for Organ H11-O10: heart; type: A+...
```

6. O órgão chega ao hospital onde o transplante será realizado.

```
Organ H11-O10: heart; type: A+ successfully arrived at Hospital 15 at Location [X=66, Y=2] for Patient H15-P1: organ: heart; type: A+; urgency: 2 by car in 0.34 hours, including a delay of 0.07 hours.
```

7. O novo órgão é transplantado com sucesso no paciente.

```
Patient H15-P1: organ: heart; type: A+; urgency: 2 was successfully implanted with Organ H11-O10: heart; type: A+.
```

5.4 Falha no transporte do órgão

1. O hospital responsável pela operação começa o processo de confirmação das condições necessárias para realizar o procedimento.

```
Hospital 3 at Location [X=11, Y=62]:
Confirming conditions for Patient H3-P16: organ: liver; type: O+; urgency: 6.
Available rooms: 4; Available teams: 1.
```

2. O hospital confirma ter as condições necessárias para a realização da operação e o transporte do órgão é iniciado.

```
Confirmation from Hospital 10 at Location [X=47, Y=55] for Patient H3-P16: organ: liver; type: O+; urgency: 6.
Initiating transport for Organ H10-O32: liver; type: AB+...
```

3. O órgão não chega ao hospital onde o transplante será realizado por falha no transporte.

```
Organ H10-O32: liver; type: AB+ failed to arrive at Hospital 3 at Location [X=11, Y=62]
```

4. Por consequência, o paciente acaba por morrer à espera de um novo órgão.

```
Patient H6-P141: organ: heart; type: A+; urgency: 1 did not find a suitable organ in time and passed away...RIP
Patient H3-P16: organ: liver; type: O+; urgency: 1 did not find a suitable organ in time and passed away...RIP
Patient H1-P54: organ: liver; type: O-; urgency: 1 did not find a suitable organ in time and passed away...RIP
Patient H3-P59: organ: liver; type: A+; urgency: 1 did not find a suitable organ in time and passed away...RIP
Patient H10-P98: organ: liver; type: B-; urgency: 1 did not find a suitable organ in time and passed away...RIP
```

5.5 Tempo de viabilidade do órgão e tempo de vida do paciente

Este teste mostra dois rins que perderam a sua viabilidade devido ao tempo que passou.

```
Organ H5-025: kidney; type: A- did not find a suitable patient in time.
Organ H7-032: kidney; type: B- did not find a suitable patient in time.
```

No próximo teste temos o caso de uma falha no transplante devido à destruição do órgão, onde depois o paciente acabou por morrer à espera de um novo órgão.

1. O hospital recusa ter as condições necessárias para a realização da operação. O sistema procura por outros hospitais para o paciente ser transplantado.

```
Conditions not met for Patient H6-P20: organ: kidney; type: A-; urgency: 5 and Organ H6-033: kidney; type: A- in Hospital 6 at Location [X=96, Y=86].
Searching for rooms in other hospitals...
```

2. O hospital alternativo confirma ter as condições necessárias para a realização da operação e o transporte do órgão e do paciente é iniciado.

```
Hospital 10 at Location [X=47, Y=55] confirmed conditions for Patient H6-P20: organ: kidney; type: A-; urgency: 5 from other hospital
0 rooms and 3 teams now available.
```

```
Conditions met for Patient H6-P20: organ: kidney; type: A-; urgency: 5 and Organ H6-033: kidney; type: A- at Hospital 15 at Location [X=66, Y=2].
Initiating patient and organ transport...
```

3. O órgão e o paciente chegam ao hospital onde o transplante será realizado.

```
Patient H6-P20: organ: kidney; type: A-; urgency: 5 arrived by car and Organ H6-033: kidney; type: A- arrived by car successfully at Hospital 10 at Location [X=47, Y=55] in 0.29 hours, including a delay of 0.0 hours.
```

4. A cirurgia falha por destruição do órgão.

```
Procedure for Patient H6-P20: organ: kidney; type: A-; urgency: 5 and Organ H6-033: kidney; type: A- failure:
Organ destroyed.
```

5. O paciente morre à espera de um novo órgão.

```
Patient H6-P20: organ: kidney; type: A-; urgency: 1 did not find a suitable organ in time and passed away...RIP
Patient H4-P211: organ: liver; type: B+; urgency: 1 did not find a suitable organ in time and passed away...RIP
```

6 Desafios e Sugestões de Melhoria

Durante o desenvolvimento do sistema, foram enfrentados diversos desafios de implementação que exigiram adaptações na abordagem para garantir o funcionamento adequado.

Um dos principais desafios foi relacionado à serialização e desserialização dos objetos. Durante esse processo, cópias dos objetos são criadas na memória, o que significa que, ao alterar um atributo de instância, é necessário garantir que a instância original seja modificada. Um exemplo disso ocorre com as instâncias de pacientes, que são serializadas e enviadas várias vezes para diferentes agentes. Quando um paciente é selecionado para

operação, é crucial impedir que ele seja novamente selecionado para transplante. Para isso, foi necessário alterar o atributo "hold" na instância do agente AT, garantindo que a instância original fosse modificada adequadamente.

Outro desafio foi a dificuldade de testar componentes isolados do sistema. Durante o desenvolvimento, observou-se que muitas funcionalidades só poderiam ser testadas efetivamente quando as dependências relacionadas também estivessem prontas, o que resultou num volume elevado de código por testar.

Diversas melhorias podem ser implementadas para tornar o sistema mais robusto e capaz de lidar de maneira mais eficiente com as complexidades e imprevistos associados ao processo de alocação de órgãos.

A primeira melhoria sugerida está relacionada à atualização do tempo de vida dos órgãos e do estado clínico dos pacientes. No sistema atual, o tempo de isquemia do órgão é contabilizado no momento da escolha do paciente, considerando o tempo de transporte. No entanto, este tempo não é atualizado enquanto o órgão está a ser transportado, o que afeta os casos em que a cirurgia não é bem-sucedida, mas o órgão permanece viável para um novo transplante. De forma similar, o nível de urgência do paciente não evolui quando este é selecionado. A proposta consiste em implementar uma atualização contínua do tempo de isquemia dos órgãos durante o transporte, bem como uma evolução dinâmica da urgência dos pacientes enquanto aguardam a operação

A segunda melhoria diz respeito à possibilidade de um paciente ser associado a múltiplos órgãos simultaneamente. Atualmente, o sistema restringe a alocação de um órgão por paciente, limitando a flexibilidade da gestão de recursos.

Outra melhoria é a possibilidade de o paciente morrer durante o transporte. Atualmente, o sistema não considera o risco de mortalidade dos pacientes críticos enquanto os órgãos estão em transporte. A proposta é introduzir uma probabilidade de mortalidade baseada na urgência do paciente. Caso o paciente morra antes de metade do trajeto, o órgão deve retornar ao hospital de origem para ser redistribuído, caso contrário o órgão fica no hospital de destino.

7 Conclusões

O sistema multiagente desenvolvido demonstrou-se eficaz na simulação de transplantes de órgãos, conseguindo gerir com sucesso o fluxo de informações entre os agentes envolvidos.

Durante o desenvolvimento foram enfrentadas algumas dificuldades e muitas vezes foi necessário recuar para reformular o raciocínio ou corrigir erros. Através da prática constante e da aplicação dos conhecimentos adquiridos ao longo da unidade curricular, foi possível superar essas dificuldades e desenvolver soluções funcionais e alinhadas aos objetivos do projeto.

Embora diversas possibilidades tenham sido exploradas durante a implementação, a verdade é que ainda muito pode ser feito com o objetivo de otimizar este sistema.