FEUP
Universidade do Porto
Faculdade de Engenharia

# Agentes e Inteligência Artificial Distribuída

Relatório Final

## Escalonamento de Pacientes em Hospitais

Mestrado Integrado em Engenharia Informática e Computação

4º ano - 1º Semestre


**Grupo T02_3:**

- André Casais Regado - ei12182@fe.up.pt
- António Presa - ei12092@fe.up.pt
- Francisco Sousa Couto - ei12189@fe.up.pt


Dezembro de 2015

# Índice

# Objectivo

O objectivo deste trabalho é a especificação e implementação de um sistema distribuído multi-agente que permita a simulação de um sistema de colocação de pacientes num hospital no âmbito da inteligência artificial.

Nesta simulação deverá existir um hospital que disponibiliza um conjunto variado de recursos (salas), distintamente geridas, que permitam tratar certos sintomas. Cada paciente deverá, à entrada no hospital ser avaliado na Triagem de modo a que lhe seja associado um "estado de saúde" que permitirá ao sistema saber o quão mal se encontra relativamente a todos os outros pacientes. Mas nem sempre o estado de saúde inicial será o final do paciente, pois poderão haver novas descobertas durante os exames, que podem mudar a prioridade dos pacientes.

Este sistema de avaliação permitirá uma gestão eficiente dos recursos presentes (que terão de ser flexíveis à mudança) que serão atribuídos, num sistema de leilão, aos intervenientes com a melhor licitação, ou seja, com estado de saúde mais deficitário.

Cada paciente deverá, caso possua mais que um sintoma, passar pelas diversas salas de que necessita sempre com o maior grau de eficácia e eficiência possível do seu ponto de vista e do ponto de vista do hospital. O objectivo final passará por evitar que, reunidas estas condições, se evite a degradação e posterior morte de cada paciente.

Para tentar resolver este problema, os hospitais começaram a usar uma técnica semelhante à *first-come-first-served*. Deste forma, uma ala do hospital receitava os exames e tratamentos a cada paciente originando *requests* para as alas auxiliares que, responsáveis por administrar os tratamentos, chamariam os pacientes conforma a sua disponibilidade no momento.

Esta técnica apesar de ser útil para responder rapidamente a mudanças repentinas e a necessidades urgentes de atendimento, não permitia uma comunicação inter-unidade o que poderia levar a que duas alas requisitassem ao mesmo tempo o mesmo paciente.

Por existir esta restrição para cada ala poderiam ocorrer tempos de espera desnecessários e desagradáveis, bem como tempos excessivos de utilização dos recursos hospitalares.

Tal como explicado anteriormente a tarefa que os hospitais têm de alocar recursos e pacientes é extremamente complexa, pelo que o objectivo seria arranjar uma solução otimizada que permitisse não só reduzir o tempo de espera de cada paciente, mas também permitir, por isso, uma alocação e utilização ótimas dos recursos disponibilizados para a aplicação de tratamentos e para a realização de exames necessários ao diagnóstico das mais variadas doenças. Para isso seria necessário implementar um sistema multi-agente e um mecanismo de coordenação que permitisse correlacionar eficientemente estes agentes.

# Especificação

## ● Identificação e caracterização dos agentes

É possível identificar três tipos de atores distintos na simulação: o hospital, os recursos desse hospital (salas) e os pacientes que querem ser atendidos, cada um representado por um agente distinto. Tal como seria de esperar a comunicação entre agentes é síncrona, cada paciente só é atendido depois de cada sala libertar o paciente que está atualmente a ser tratado. Segue-se a especificação dos agentes implementados:

AgenteHospital:

Este agente é representativo como um conjunto de salas que serão solicitadas por cada paciente dependente do(s) seu(s) sintoma(s). Possui uma *GUI* de forma a tornar a interoperabilidade com o utilizador mais fluída e simples.

AgentePaciente:

Parâmetros:

- Sintomas [febre,calos]: Conjunto de variáveis que servirão para caracterizar o estado de saúde do agente. A partir delas descobriremos as salas que será necessário visitar de modo a que se tente melhorar o seu estado de saúde e evitar que morra/piore.

Este agente representa cada paciente que deseja visitar o hospital. É caracterizado por um conjunto de sintomas. O seu *lifecycle* será delineado pela visita a várias salas que possuem a competência para tratar os seus sintomas e ajudar a melhor o seu estado de saúde. Este estado será um valor decimal entre 0,0 (morte) e 1.0 (completamente tratado) e será,por isso, mutável ao longo do tempo.

É representado por dois behaviour's, o RequestCheckUp e ResquestPerformer. No primeiro behaviour (RequestCheckUp) o paciente, procura pela sala cujo o nome é triagem e envia uma mensagem cujo conteúdo é os seus sintomas e de seguida, recebe uma resposta com o seu estado de saúde de acordo com a analise da triagem perante os seus sintomas. O segundo behaviour (RequestPerformer) é responsável por receber uma mensagem da sala onde têm que ir, com a informação de se dirigir a ela, mandando o seu estado de saúde atual ao recurso. De seguida, recebe o resultado do exame efetuado, recebendo o seu estado de saúde novo, consoante esse novo estado , tem o seu efeito no paciente.

AgenteRecurso

Cada agente deste tipo é criada pela *GUI* associada ao hospital que detém esses recursos. Podem existir seis tipos de recursos disponíveis: Oncologia, Pediatria, Urgência, Ortopedia, Genecologia, Medicina Dentária. Cada sala poderá atender apenas um paciente de cada vez e dará prioridade consoante o seu estado atual de saúde. A forma como essa avaliação poderá mudar dependerá de sala para sala, visto que pacientes que necessitem de tratamento nas salas de Oncologia e Urgência terão maior probabilidade de piorarem ou melhorarem muito pouco. Por outro lado, agentes que solicitem atendimento na Ortopedia, por exemplo, muito provavelmente não terão complicações no seu tratamento e poderão ser facilmente atendidos e tratados com sucesso.

É também representado por dois behaviour's, CheckUp e OfferRequestServer. O primeiro behaviour (CheckUp) tem um comportamento cyclicbehaviour, em que recebe os sintomas de um pacientes, determinando assim , os recursos que os pacientes precisam de ir e calcula o seu estado de saúde, enviando depois ao paciente o seu estado de saúde. O segundo behaviour (OfferRequestServer) também com um comportamento cyclicbehaviour , em que escolhe o paciente com pior estado de saúde que necessita de ir aquele recurso mas só se estiver disponível se não escolhe o próximo com estado de saúde menor. Depois de escolhido o paciente com menor estado de saúde menor e disponível é lhe mandado uma mensagem para vir à sala. Depois do exame, manda a esse mesmo paciente o resultado e o seu novo estado de saúde. Por fim, termina a sua conexão a esse paciente.

- Protocolo de Interação

Sendo o sistema síncrono, pode ser especificado por uma lista de mensagens trocadas entre os agentes, da seguinte forma:

1. O paciente envia os sintomas ao hospital, que o envia à triagem para calcular o seu estado de saúde.
2. O hospital, consoante esses sintomas, avalia o estado de saúdo do paciente e envia esse estado ao paciente.
3. A sala/recurso, quando disponível, envia uma mensagem ao paciente que está em primeiro na sua lista (consoante o seu estado de saúde, ou seja, o paciente que esteja pior), que esteja disponível.
4. Depois de fazer o exame dessa sala, o paciente é reavaliado, e a sala manda mensagem com o seu novo estado de saúde ao paciente.
5. Consoante o resultado, o paciente ou morre, ou porque melhorou abandona o hospital ou então se piorou mas não morreu, volta a fazer de novo o exame.

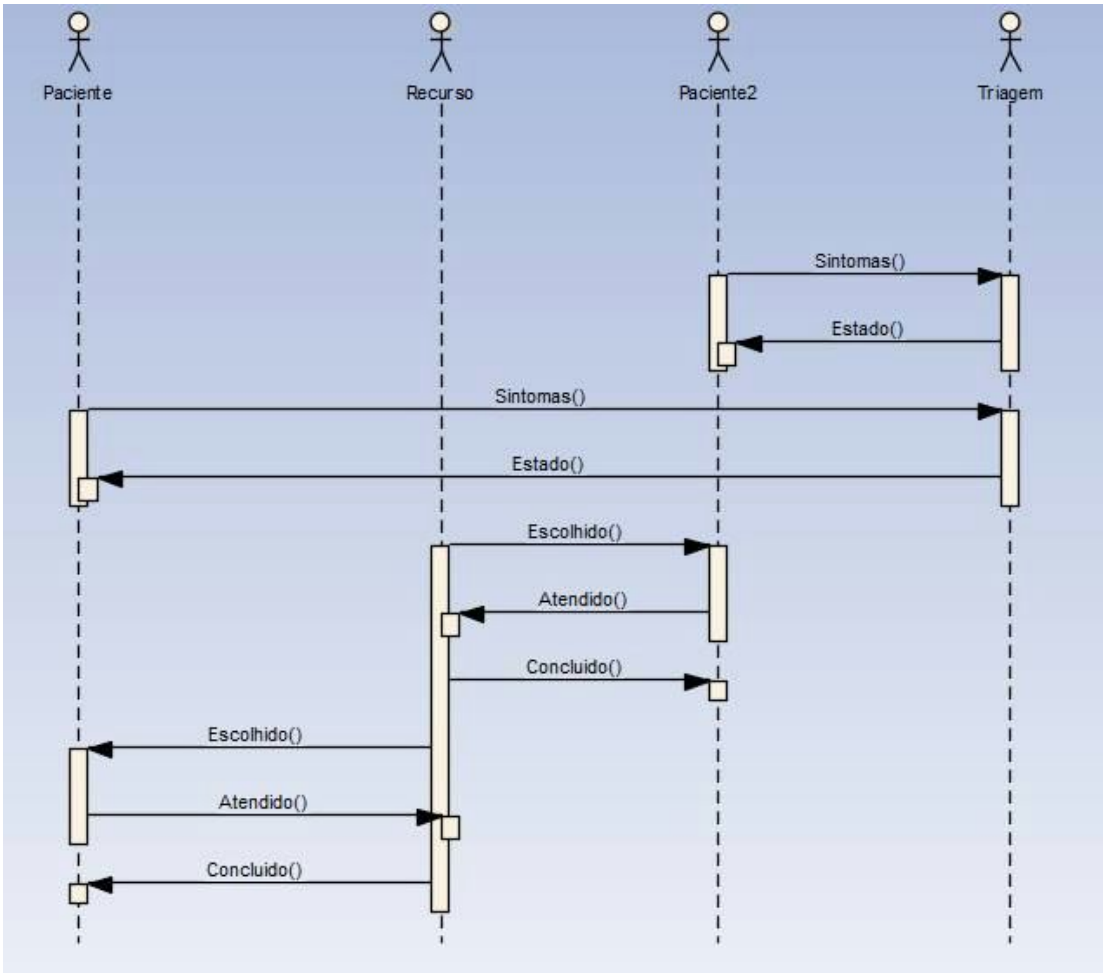



Figura 1 - Diagrama de sequência

# Desenvolvimento

O trabalho foi desenvolvido em Java, usando o Eclipse como IDE em máquinas com *windows* 8.1 e 10. O sistema foi desenvolvido com recurso à *framework* JADE (*Java Agent Development Framework*) que permite a implementação de sistemas multi-agente seguindo os standards especificados pela *FIPA* e disponibiliza algumas ferramentas que ajudam ao *debugging* e *deployment* das várias fases do projecto.

Esta *framework* permite ainda a distribuição do sistema por várias máquinas sem necessidade de partilharem o mesmo sistema operativo.

Alguma informação importante, mas temporária, é adicionada/editada/removida em ficheiros de texto de forma a auxiliar as escolhas feitas pelos vários agentes ao longo do tempo, nomeadamente no que diz respeito à atualização dos estados e à escolha dos mesmos por parte dos recursos. Estes ficheiros adotam o nome do agente que os solicita (ex: Oncologia.txt) e surgem no formato nomeAgente;Estado de forma a facilitar a sua interpretação.

O sistema implementa uma arquitetura de sistemas comunicativos e o desenrolar das licitações e tratamentos deve-se à correta comunicação entre eles e consequentes modificações dos vários parâmetros pertinentes ao funcionamento dos leilões (por exemplo). As mensagens são distinguidas pela sua performativa e pelo id da conversa em curso. Visto o sistema ser síncrono é de esperar que se consiga distinguir, a cada momento, o tipo de mensagem a receber por todos os agentes criados e em ação naquele momento. Dessa mesma forma é, por isso, possível verificar o correto funcionamento da asserção de prioridades, bem como da atualização dos estados de saúdes visto serem diretamente dependentes das mensagens, algo que é facilitado pela *GUI existente*.

Concluindo, o sistema é composto pelos diversos agentes e respectivos *behaviours*. O hospital é o agente principal, visto, sem este, não ser possível existirem salas e consequentemente, tratamentos.

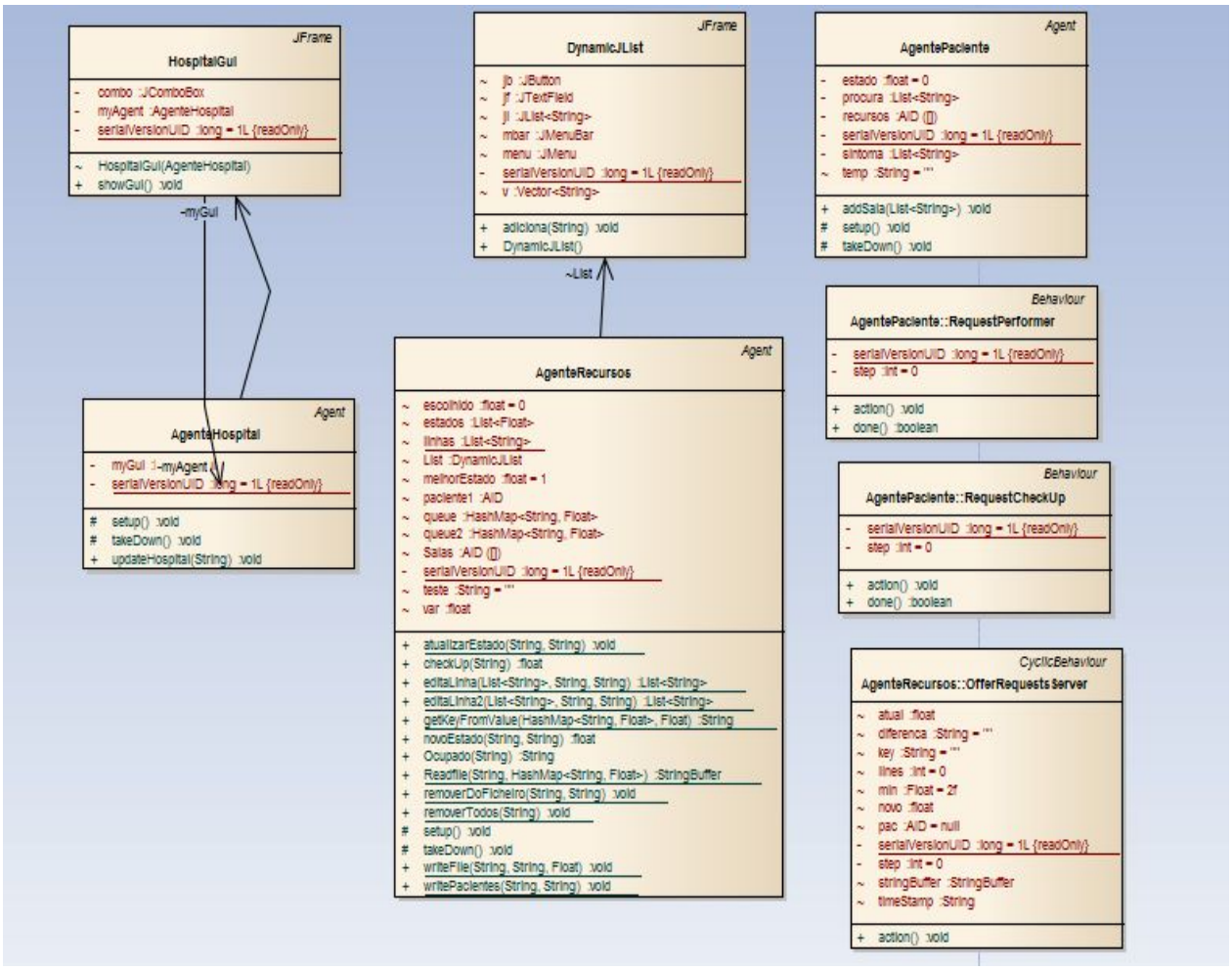
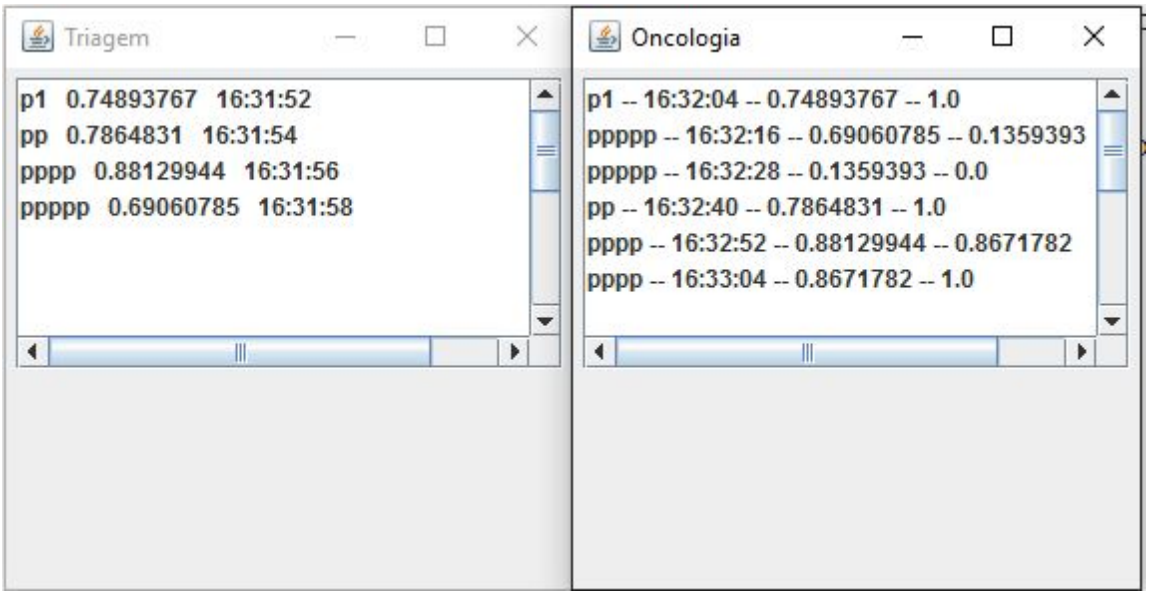



Figura 2 -Diagrama de classes do sistema.

# Experiências

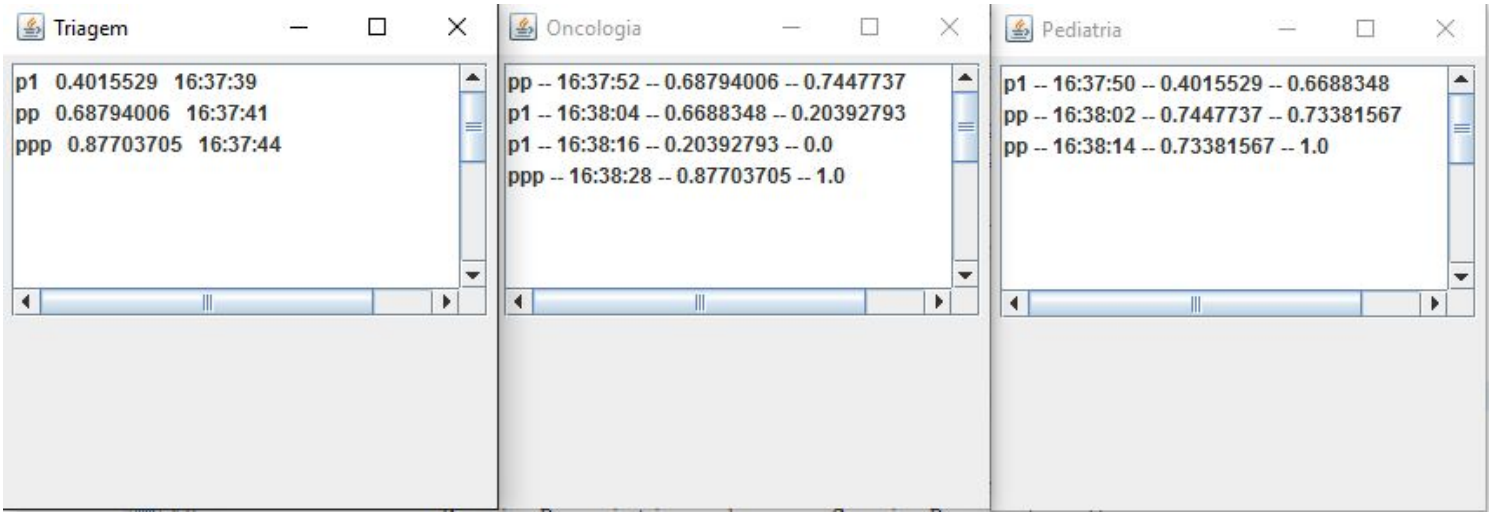
Tendo em conta os objetivos do trabalho desenvolvido poderão ser postas em práticas três experiências diferentes:

1. Criar vários pacientes para a mesma sala e observar como progressivamente é escolhido o que está mais débil em termos de saúde;




Através desta experiência, percebemos que os pacientes são atendidos de forma correta, ou seja, a partir do mais "doente". O paciente *p1*, chegou e a sala como estava livre , entrou logo, de seguida enquanto o paciente *p1* está a ser atendido chega os pacientes *p1*, *pp*, *pppp*, *ppppp*, cujos os seus estados de saúde são aproximadamente 0.78, 0.88 e 0.69 respectivamente. Depois do *p1* ser atendido (correu bem o exame pois ficou curado e regressou a casa), houve um leilão entre os outros 3 pacientes, mas como o paciente *ppppp* estava com um estado de saúde menor, entrou ele, mas no 1º exame piorou bastante, e continuando ser o paciente com mais prioridade, voltou outra vez a sala mas acabou por morrer (estado de saúde 0). De seguida, só há mais 2 pacientes, mas o *pp* tem o estado de saúde menor, logo é ele atendido mas também analisamos que correu mal e acabou por morrer. Por fim, foi atendido o paciente *ppppp*, o primeiro exame correu um pouco mal, pois piorou o seu estado de saúde, mas no próximo exame , melhorou e foi enviado embora.

2. Criar mais do que uma sala e vários pacientes para serem alocados, verificar atendimento síncrono, bem como a escolha, em cada sala, do melhor paciente com o pior estado; Existindo um paciente que necessite de mais que uma sala verificar a atualização do seu estado de saúde;




Sendo que todos os pacientes entraram com sintomas que necessitavam de fazer exames nas salas de oncologia e pediatria, reparamos que o que tem menor estado de saúde (*p1*), é escolhido para ir para a pediatria, enquanto está na pediatria, o paciente com o 2º menor estado de saúde (*pp*) foi atendido na sala da oncologia. Tanto o paciente *p* e *p1* melhoraram um pouco o seu estado de saúde, por isso, abandonaram aquela sala. Depois como, mesmo assim, tinham estado de saúde menor que o outro paciente (*ppp*) trocaram de sala, ou sejam o *p1* foi atendido na oncologia e o *pp* na pediatria. Por fim, a primeira sala a ser libertada, selecionou , neste caso, o último paciente *ppp*. Neste caso sou a sala da oncologia a ficar livre mais rápido e teve a felicidade de o exame correr bem e como ficou totalmente curado (estado de saúde = 1), não necessita de ir à outra sala.

# Conclusões

Levando a cabo as experiências e a partir do desenvolvimento do projecto verificamos que de facto a aplicação de um sistema multi agente é uma excelente resolução para este problema de escalonamento, visto facilitar tremendamente a aplicação de uma troca de mensagens entre duas ou mais entidades de forma sucessiva.

Vejamos o exemplo da experiência dois em que possuímos múltiplos agentes de cada tipo, verificamos a enorme maleabilidade que um sistema deste gênero desenvolvido com JADE possui, permitindo que de forma independente, cada recurso interaja com os pacientes existentes, criando-se assim uma interoperabilidade entre todo o sistema.

Isto permite que cada paciente veja o seu tempo de espera extremamente encurtado visto que, por um lado, existe uma escolha inteligente na seleção de cada paciente alocado a cada recurso e, por outro, nenhuma sala está dependente de outra para funcionar, na medida em que se um paciente X requisitar a sala Y e a sala Z, mas apenas ganhar a licitação na Y, a Z não fica a aguardar por este, mas seleciona outro paciente. Isto é extremamente importante e demonstra um enorme upgrade em relação à estratégia que os hospitais implementam, tal como referido, a *first come, first served*.

Por outro lado, é importante realçar que um projecto deste genero convive com a realidade de variáveis um pouco abstratas no que diz respeito ao estado de avaliação de cada paciente, por exemplo, na medida em que a forma como este melhora ou piora em cada estágio do seu tratamento é sempre subjectivo, mas é uma necessidade que exista de forma a poder existir uma progressão na execução do sistema.

Com este projeto, o grupo pensa, que conseguiu resolver os problemas das técnicas dantes usadas pelo hospital, como por exemplo o first-come first-served. Asseguramos assim, que as salas/recursos, sempre que haja pacientes, estão em constante utilização. Que mesmo que um paciente seja chamado ao mesmo tempo para duas salas, ele dirige-se para uma e a outra chama o próximo paciente com pior estado de saúde que requer aquela sala, evitando assim o que acontecia anteriormente, a inutilização dos recursos.

Concluindo, o objetivo deste trabalho foram concretizados, rentabilizar a utilização dos recursos minimizando os tempos de inutilização e os pacientes minimizar o tempo de permanência no hospital.

# Melhoramentos

Agilizar o processo de troca de mensagens usando possivelmente outros recursos para ajudar à sincronização e atualização das mesmas, neste caso no que diz respeito, principalmente, aos estados de cada paciente e à sua possibilidade de ser atendido.

Seria interessante também, o grupo ter implementado a hipótese de correr o projeto com outras estratégia como por exemplo, *first-come first-served*. Podendo assim comparar os resultados entre estes dois tipos de abordagem e perceber realmente o quanto a abordagem implementada pelo grupo, é melhor. Mas devido a falta de tempo, não foi possível.

# Recursos

- ## Bibliografia


1. Paulussen, T., Zöller, A., Rothlauf, F., Heinzl, A., Braubach, L., Pokahr, A., Lamersdorf, W.: Agent-based patient scheduling in hospitals. In: P.L.O.S. S. Kirn O. Herzog (ed.) Multiagent Engineering - Theory and Applications in Enterprises, pp. 255-275. Springer (2006)
2. Braubach, L. , Pokahr, A. , Lamersdorf, W.:Negotiation-based Patient Scheduling in Hospitals (https://www.activecomponents.org/bin/download/Documentation/Publications/medpage.pdf)
3. https://www.activecomponents.org/bin/download/Documentation/Publications/medpage.pdf https://vsis-www.informatik.uni-hamburg.de/getDoc.php/publications/289/13_060111-III.4-PaulussenZoellerRothlaufHeinzlBraubachPokahrLamersdorf-FINAL.pdf (12/12/2015)

- Software

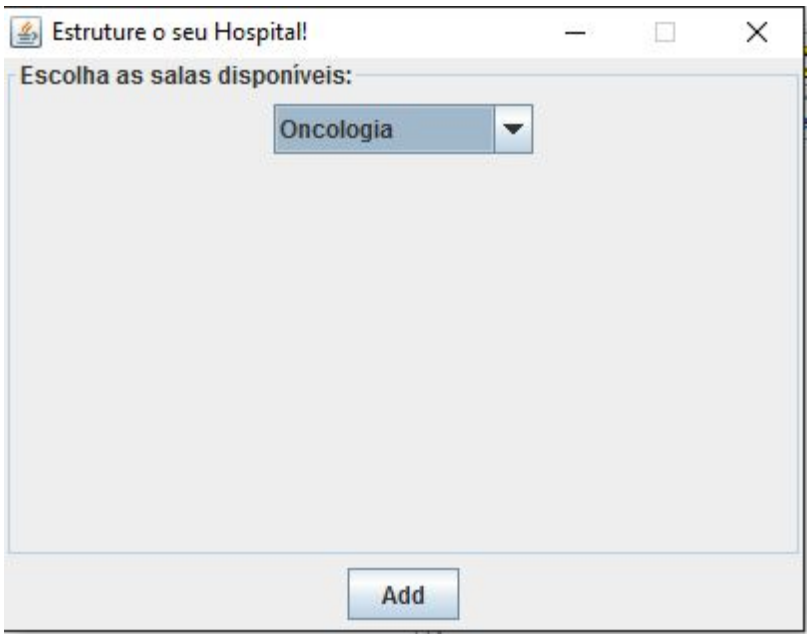
Jade - http://jade.tilab.com/

- Contribuição dos elementos do grupo

Todos os elementos contribuíram igualmente para o desenvolvimento do trabalho visto que este foi feito maioritariamente em conjunto, analisando e discutindo.

# Apêndice

De modo a ser utilizado o programa desenvolvido, correndo a aplicação, o utilizador deverá criar um agente hospital utilizando a *GUI* do Jade. Uma interface criada pelo grupo abrirá de forma a que se adicione as salas que pretendemos, Oncologia, Pediatria, etc.




Criadas as salas poderemos começar a lançar os agente Paciente pretendidos, não esquecendo a passagem, como argumentos, dos vários sintomas. Para facilitar a interação

e para torná-la mais rápida, os sintomas são “cancro”, “febre”, “calos”, “baleado”, “gravidez” e “caries”. Estes devem ser, se existirem vários, separados por vírgula. Como não fazia sentido para a simulação, um agente sem qualquer parâmetro será imediatamente apagado do sistema.


Criados os pacientes o utilizador poderá, a partir da *GUI* de cada sala, verificar o desenrolar do programa. Dessa forma, começará por verificar o atendimento dos clientes na triagem onde estes fazem o seu *checkup* e são avaliados. Posteriormente e de forma sequencial verificará o atendimento nas diversas salas dos vários agentes podendo consultar a hora a que este atendimento ocorreu, bem como o estado de saúde com que chegou e aquele com que saiu.