UNIVERSIDADE DO MINHO

AGENTES INTELIGENTES

SISTEMA AGENTE E MULTIAGENTE

Grupo:
João Nunes - A82300
Mateus Silva - A81952

Braga, Portugal
4 de Novembro de 2019

Conteúdo

Lista de Figuras

1 Introdução

Neste presente relatório, será descrito todo o processo de concessão e de estruturação do trabalho prático da unidade curricular de Agentes Inteligentes que se foca na criação de um sistema de agentes inteligentes com o foco de apagar incêndios.

O principal objetivo do trabalho é agilizar e maximizar a efectividade de um sistema de mobilização de meios para apagar os eventuais fogos existentes pela grelha fora. Como tal e de modo a possuir algumas bases e uma melhor compreensão acerca do assunto foi necessário desenvolver alguns diagramas em UML (Unified Modeling Language) dos quais se destacam o modelo lógico, nos quais se identificam as entidades do sistema e os seus atributos, o diagrama de classes sendo esta a evolução do modelo lógico já com as classes bem identificadas, os diagramas de sequência dos quais se destacam alguns métodos principais do sistema bem como algumas formas de comunicação, os diagramas de atividade e para finalizar uma máquina de estado.

2 Estado de arte

2.1 Sistemas multiagentes

Um sistema multiagente carateriza-se como sendo um sistema em que diversas unidades computacionais interagem entre si e perseguem um conjunto de objetivos realizando tarefas para os atingir. Portanto, um agente corporiza um sistema computacional capaz de revelar um ação autónoma e flexível, possuindo uma existência própria pelo que são independentes da existência de outros agentes. Portanto a decisão de qual ação a levar avante é determinada pelo agente tendo em consideração as mudanças que ocorrem no seu ambiente bem como os seus objetivos.

Como tal, e no âmbito do trabalho proposto, será criado um sistema multiagente aplicado na área de simulação de combates a incêndios florestais através do uso de agentes cujo intuito é apagar os fogos, sendo estes agentes coordenados através do agente central. Existe também o agente incendiário cujo objetivo é atear os fogos.

Escusado será dizer portanto a importância de desenvolvimento de aplicações que permitam simular a ocorrência de incêndios, uma vez que com estas ferramentas é possível simular situações potencialmente reais e tirar conclusões partindo destas simulações. Por exemplo, num projeto de larga escala, seria possível definir uma topologia (mapa) na simulação mais próxima da realidade onde seria possível então maximizar a eficácia na distribuição de meios para apagar os fogos tendo em conta a topologia da região em questão definida no mapa, podendo estes conhecimentos ser empregues potencialmente na realidade, ou pelo menos fornecer dados suficientes para melhorar o processo real.

Este cenário poderá ser implementado através do uso de uma *framework* para o JAVA denominado de *JADE* (Java Agent Development) que é uma plataforma de agentes com o objetivo de simplificar o desenvolvimento deste tipo de sistemas, proporcionado também uma API gráfica com o intuito de facilitar a administração deste tipo de sistemas. A base da comunicação proporcionada pelo *JADE* é garantida pela *ACL* (Agent Communication Language) que obedece à norma internacional *FIPA*, as mensagens trocadas entre os agentes poderão ser de diferentes tipos, sendo estes tipos definidos através de *PERFORMATIVES* que poderão variar deste propostas, pedidos ou até mesmo mensagens de informação. No que toca aos agentes em si, estes poderão exibir diferentes tipos de comportamento, *behaviours*, consoante o objetivo que se pretende.

2.2 Aplicações reais

No que toca a casos concretos de utilização do JADE e mais aplicados ao âmbito deste trabalho, existe um artigo[1] que pretende explicar a proatividade dos agentes ao invês de estes serem apenas reativos, usando para tal um contexto semelhante ao tema do trabalho proposto que é combate aos incêndios urbanos, relativamente à coordenação e distribuição de tarefas de agentes num ambiente simulado de incêndios florestais existe algum desenvolvimento nesta área no qual se pode destacar o *Fighting Fire with Agents*[2], *Fighting fire with fear*[3], *Coordinating a team of agents in the forest firefighting domain*[4] e por fim *Strategies for Route Planning on Catastrophe Environments*[5].

Relativamente a outros projetos em que os sistemas multiagente são implementados poderá se destacar o uso deste tipo de sistemas na gestão de redes de energia[6], os sistemas multiagente também podem ser implementados em contextos lúdicos[7] como por exemplo em jogos estratégicos de formação de palavras onde os agentes interagem entre si para obter as letras necessárias para formar a palavra pretendida.

Portanto, e como é possível observar os *SMA* podem ser aplicados em diversas áreas, com diferentes contextos e vertentes, sendo que no seu cerne é ideal para ser aplicado em contextos de simulação.

3 Arquitetura da aplicação

3.1 Modelo de domínio

Tendo como principal objetivo a elaboração de um sistema multiagente num cenário de combate de fogos, e após algum tempo despendido a analisar o enunciado foi possível levantar algumas entidades e alguns atributos, sendo que o resultado do levantamento de entidade se encontra de seguida.

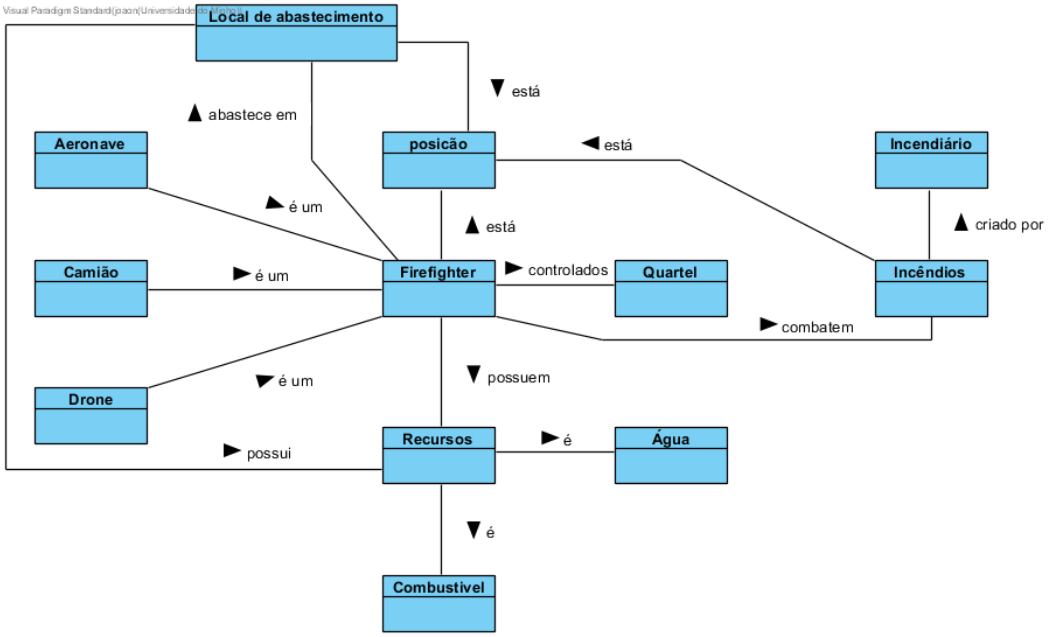

Figura 3.1: Modelo de domínio.

3.1.1 Entidades relevantes

Os meios que combatem os fogos , sendo estes denominados por **firefighters** que poderão ser uma **aeronave**, **camião** ou até mesmo um **drone**. Estes possuem recursos como água/combustível, e para além disso também estarão a associados a estes uma posição na grelha. Estes meios são controlados por um **quartel** que irá indicar a célula para onde se devem mover para apagar um eventual fogo ou para abastecer.

Para além disso, é ainda relevante a entidade **incendiário** que dita a incidência de fogos que acontecerão pela grelha fora, sendo para tal necessário também uma posição que corresponde à posição na grelha em que o fogo está a acontecer. Um **local de abastecimento** também é uma entidade do sistema, uma vez que durante o processo de apagar um dado fogo, são consumidos

recursos, o combustível que corresponde ao custo de se mover até à célula onde existe o fogo e também se consome água que é o custo de apagar o fogo. Como tal, também é necessário coordenar os meios de modo a abastecer os seus recursos.

3.1.2 Subsistemas principais

Analisando o modelo de domínio anterior, é possível inferir que a aplicação poderá ser dividida em dois subsistemas principais:

1. Subsistema agente *firefighter*;
2. Subsistema agente quartel.

Estes dois subsistemas englobam a maior parte e o grosso de todo o sistema, sendo que a maior parte das funcionalidades se irão encontrar concentradas nestes dois subsistemas principais.

O subsistema agente *firefighter* de uma maneira geral é responsável por guardar a maior parte da informação referente aos meios de combate aos incêndios seja isso a capacidade atual de água, a sua velocidade entre outros. Irá também possuir algumas funcionalidades como enviar a informação atual para o quartel de bombeiros, mover para uma certa posição na grelha e entre outros que irão ser abordados no capítulo seguinte.

Por sua vez o subsistema agente quartel é responsável pela gestão e dissipação dos meios de combate aos incêndio pelo mapa, sendo que, por exemplo, a informação de ocorrência dos incêndios passará por este subsistema. No que toca as suas funcionalidades, este terá que estabelecer uma troca de informação com os meios de combate aos incêndios, ou seja, os *firefighters* sendo para tal necessário receber a informação desses meios, e também deverá poder ordenar um *firefighter* para se mover para um certa posição na grelha.

4 Diagrama de classes

4.1 Diagrama

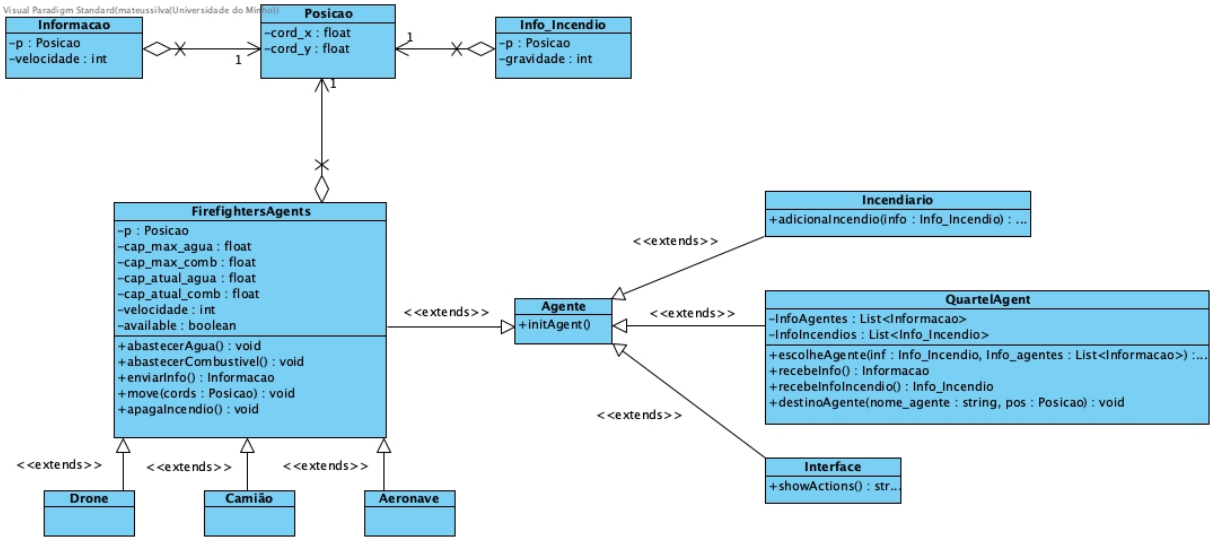

Figura 4.1: Diagrama de classes

Após uma melhor análise sobre as entidades envolvidas no problema em questão, foi possível desenvolver um modelo de classes onde é já possível identificar as classes principais do sistema, bem como as suas relações, variáveis e métodos, que mais tarde podem ser vistos como *behaviors* dos agentes.

Neste diagrama podemos identificar como classes principais os *FirefightersAgents*, *Incendiario*, *QuartelAgent* e *Interface*. Estas classes irão ser os futuros agentes do sistema. As classes referidas dão extends à classe *Agent* que tem o método *initAgent* que serve para inicializar os agentes. O Drone, Camião e Aeronave são tipos de *FirefightersAgents*, e têm como variáveis de instância uma *Posicao* (posição em que se encontra), *cap_max_agua* (capacidade máxima de água), *cap_max_comb* (capacidade máxima de combustível), *cap_atual_agua* (capacidade atual de água), *cap_atual_comb* (capacidade atual de combustível), *velocidade* (velocidade a que se consegue mover um agente) e *available* (um boolean que diz se o agente está disponível). Como métodos têm o *abastecerAgua*, *abastecerCombustivel*, *enviarInfo*, *move* e *apaga*, que tal como os nomes intuitivamente demonstram, definem o comportamento destes agentes.

O *QuartelAgent* possui como variáveis duas listas, uma para guardar a informação dos agentes, *InfoAgents*, e outra para guardar a informação dos incêndios, *InfoIncendios*. Dentro dos métodos temos *escolheAgente*, *recebeInfo*, *recebeInfoIncendio* e *destinoAgente*.

A *Interface* tem apenas a função de mostrar graficamente o estado atual do sistema.

O *Incendiário* é a classe que define o agente incendiário, tendo este como principal função a criação de incêndios, sendo estes caracterizados por uma *Posicao* e por uma *gravidade*.

Como classes auxiliares temos a *Posicao*, que tem as coordenadas x e y, a *Informacao*, que serve para guardar a informação relevante de um *FireFighterAgent* (instâncias desta classe vão ser mandadas para o *QuartelAgent* quando este necessitar de saber informação) e *Info_Incendio*, que guarda a informação de um incêndio (instâncias desta classe vão ser mandadas do agente *Incendiário* para o agente *QuartelAgent* para informar de um novo incêndio).

É também importante saber o funcionamento geral da aplicação tendo por base estas classes. O agente Incendiário cria um novo objeto *Info_Incendio* que contem a informação de um incêndio e manda para o *QuartelAgent* usando o método *adicionaIncendio*. O *QuartelAgent* recebe o objeto *Info_Incendio* usando para isso o método *recebeInfoIncendio*. Após isto, o *QuartelAgent* vai pedir informação aos *FireFighters*, sendo que estes criam um objeto *Informacao* com a sua informação atual, enviando-a usando o método *enviarInfo*. Estas instâncias de *Informacao* enviados são recebidos no *QuartelAgent* usando o método *recebeInfo*, e tendo agora informação de todos os agentes, este decide qual o melhor para tratar o incêndio atual usando o método *escolheAgent*. Esta escolha baseia-se na *Posicao* e na *velocidade* de cada agente. Após a escolha do agente, o *QuartelAgent* informa o respetivo *FireFightersAgent* usando o método *destinoAgente*. Após receber a mensagem o *FireFightersAgent* tem de verificar inúmeras variáveis para testar a sua possibilidade de ir apagar o incêndio, nomeadamente se tem combustível e água suficientes para apagar o incêndio. Depois desta verificação o agente informa o *QuartelAgent*.

5 Diagramas de sequência

De modo a assistir e a facilitar o conjunto de interações que os interveniente no sistema terão entre si, foram criado alguns diagramas de sequência, apenas os principais, tentando detalhar e aprofundar as principais funcionalidades que o sistema deverá ter, tendo como base o diagrama de classes criado anteriormente.

De resto, basta realçar que a comunicação inter-agente tem por base a especificação e estrutura da norma *FIPA Agent Communication Language (ACL)* tendo sido associadas às mensagens algum tipo de *performative* bem como alguma estruturação do formato e conteúdo na troca de mensagens.

5.1 Enviar Informação

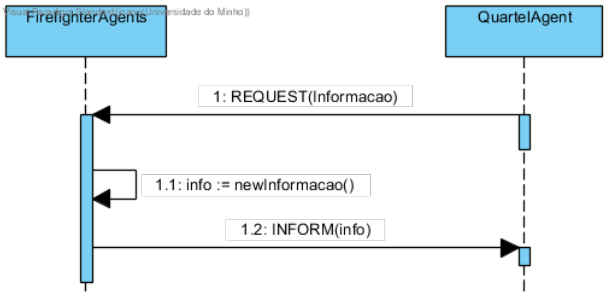

Figura 5.1: Diagrama de sequência do enviar informação.

No que toca ao envio de informação entre os agentes *firefighters* com o quartel, o quartel terá que fazer um *request* de modo a pedir que estes lhe mandem a informação atual, de modo a manter uma lista com informação coerente e atualizada. Como tal no agente *firefighter* será necessário criar um objeto do tipo informação que possuirá a sua posição atual e a sua velocidade, estes dados são depois embebidos num *serialized object* com o objetivo de colocar estes dados como conteúdo numa mensagem para o agente central.

5.2 Adiciona Incêndio

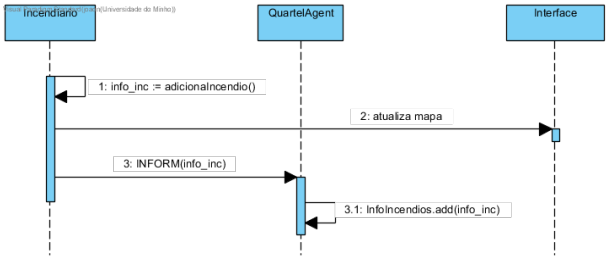

Figura 5.2: Diagrama de sequência do adicionar incêndio na grelha.

O agente incendiário é responsável pela ocorrência dos focos de incêndios pela grelha fora, como tal é necessário estabelecer a comunicação entre este e o quartel, de modo a permitir que os agentes *firefighters* possam apagar os fogos.

Como tal, o incendiário irá começar o fogo numa posição (x,y) e com uma certa gravidade, estes dados irão de seguida ser encapsulados num objeto denominado *Info_Incendio*, é também necessário comunicar esta nova alteração com a interface de modo a atualizar o seu estado, de seguida esta mensagem é comunicada com o agente quartel através da *performative INFORM* sendo o conteudo da mensagem o *serialized object* correspondente ao *Info_Incendio* criado anteriormente. No agente quartel esta nova ocorrência do fogo será adicionada a uma lista de incêndios que é mantida neste agente.

5.3 Escolhe Agente

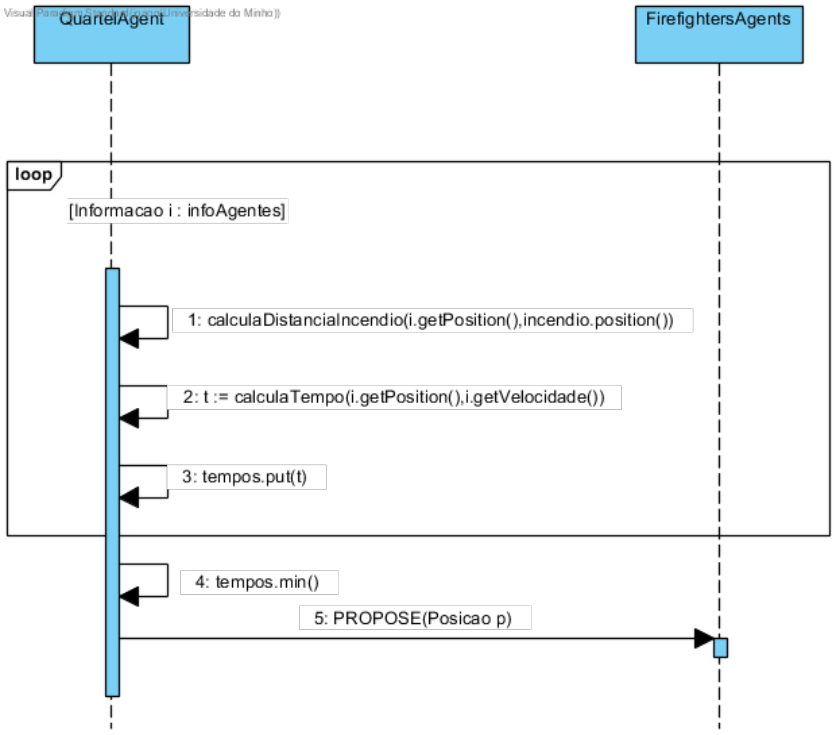

Figura 5.3: Diagrama de sequência para escolher agente.

Na escolha de um certo agente para apagar um incêndio é primeiro necessário percorrer a lista de informação e calcular a distância de cada agente do sistema à posição do fogo, de seguida e com base nesta distância é também necessário conjugar esta com a velocidade do agente, pelo que é também necessário calcular a velocidade do agente para percorrer essa dada distância com base na sua posição inicial e a sua velocidade. Por fim este tempo é posto numa lista sendo escolhido o agente que possuir o menor tempo, no final é enviada para o *firefighter agent* escolhido uma mensagem do quartel com a *performative PROPOSE* contendo um objeto *serializable* com a posição do incêndio e o tempo estimado para chegar ao incêndio.

5.4 Apagar Incêndio

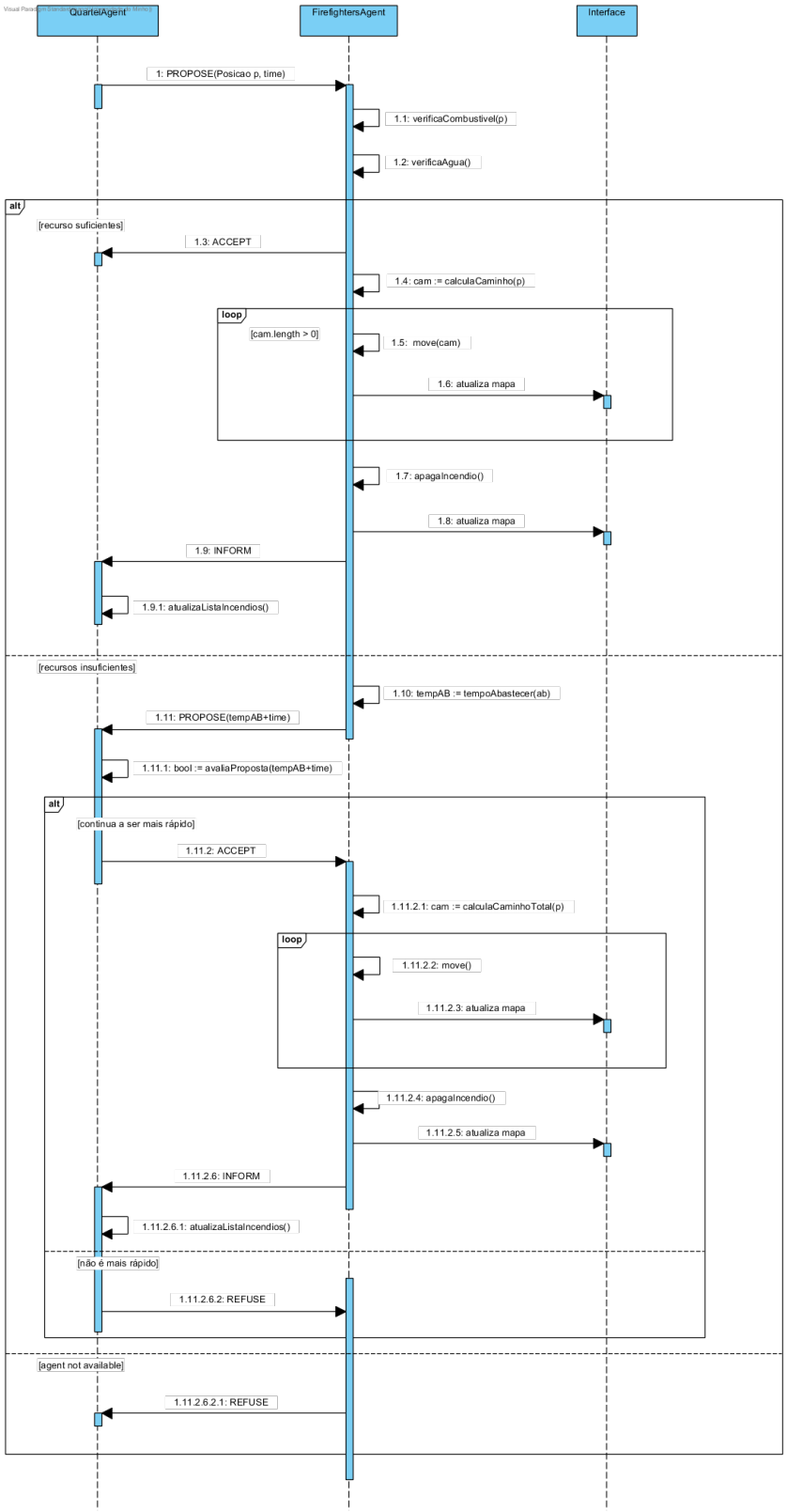

Figura 5.4: Diagrama de sequência para agente apagar incêndio.

Após uma análise sobre o *firefighter agent* mais rápido através do escolhe agente apresentado anteriormente, o quartel agente irá comunicar com esse agente fazendo uma mensagem com a *performative PROPOSE* onde irá enviar um objeto *serializable* com a posição do incêndio bem como o tempo estimado que demorará a chegar a essa posição. De seguida o *firefighter agent* recebe a proposta com esse objeto e irá analisar os seus recursos.

O *firefighter agent* irá possuir três possíveis respostas a este *PROPOSE*, dependem dos seus recursos e do seu estado. Caso possua recursos suficientes para apagar o incêndio no tempo previsto pelo quartel este irá responder com um *performative ACCEPT* para o quartel e irá apagar o fogo, começando por calcular o caminho a percorrer. Caso tenha recursos insuficientes irá ocorrer uma contraproposta através de uma *performative PROPOSE* que contém o tempo de abastecimento mais o tempo de chegada ao incêndio, o *agent quartel* irá avaliar esta proposta respondendo de duas formas. Se a proposta do agente continuar a ser a mais rápida irá responder com a *performative ACCEPT* e o *firefighter agent* terá que calcular o caminho total até chegar ao local do incêndio e o apagar. Caso o tempo calculado seja pior do que outro agente o *quartel agent* irá responder com a *performative REFUSE*.

O último caso contemplado ocorre quando o *firefighter agent* já está delegado a outro incêndio pelo que irá recusar a proposta original feita pelo *agent quartel* com a *performative REFUSE*.

6 Diagramas de atividade

De modo a complementar os diagramas de sequência criados anteriormente, foram criados algums diagramas de atividade que melhor representam o fluxo de trabalho do sistema idealizado pelo grupo, desta maneira é de esperar que seja mais clara a interação entre os diferentes agentes no sistema.

Os diagramas de atividade foram feitos tendo por base os diagramas de sequência criados anteriormente, pelo que a explicação dada nos diagramas do capítulo anterior ainda se mantêm para este tipo de diagramas.

6.1 Escolhe Agente

s

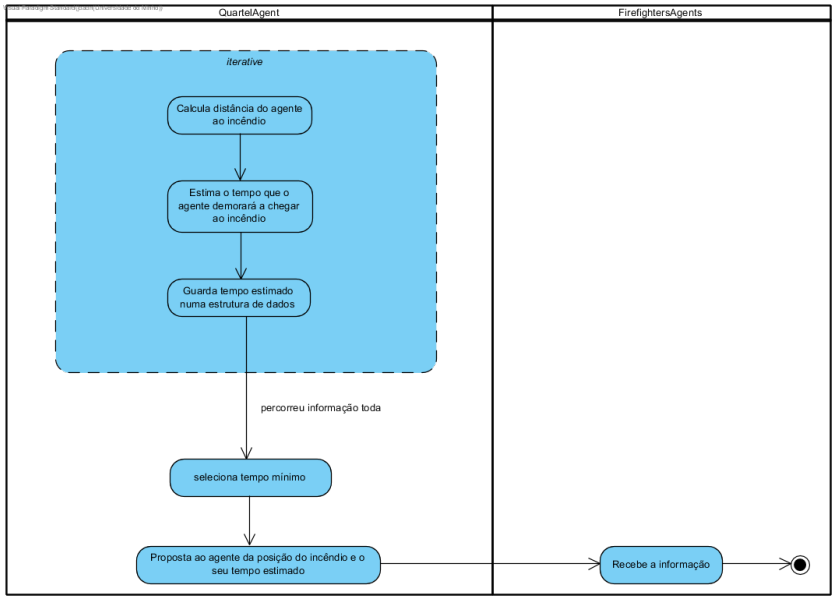

Figura 6.1: Diagrama de atividade para escolher agente.

6.2 Enviar Informação

s

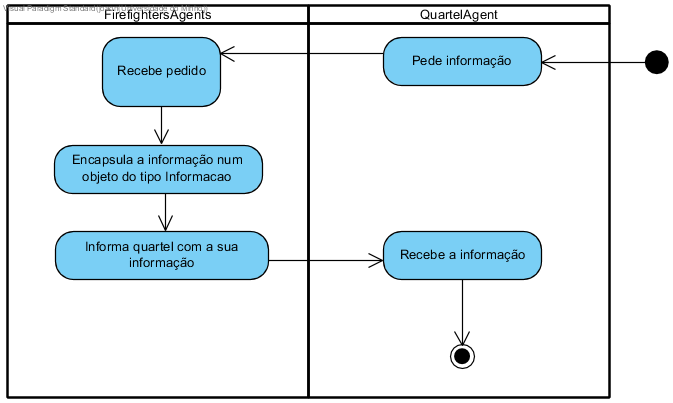

Figura 6.2: Diagrama de atividade do enviar informação.

6.3 Adiciona Incêndio

s

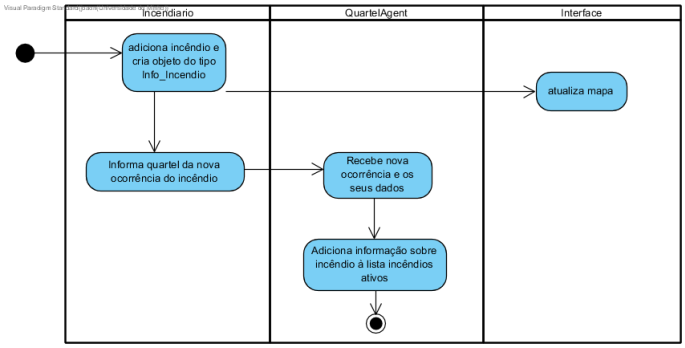

Figura 6.3: Diagrama de atividade do adicionar incêndio na grelha.

6.4 Apagar Incêndio

s

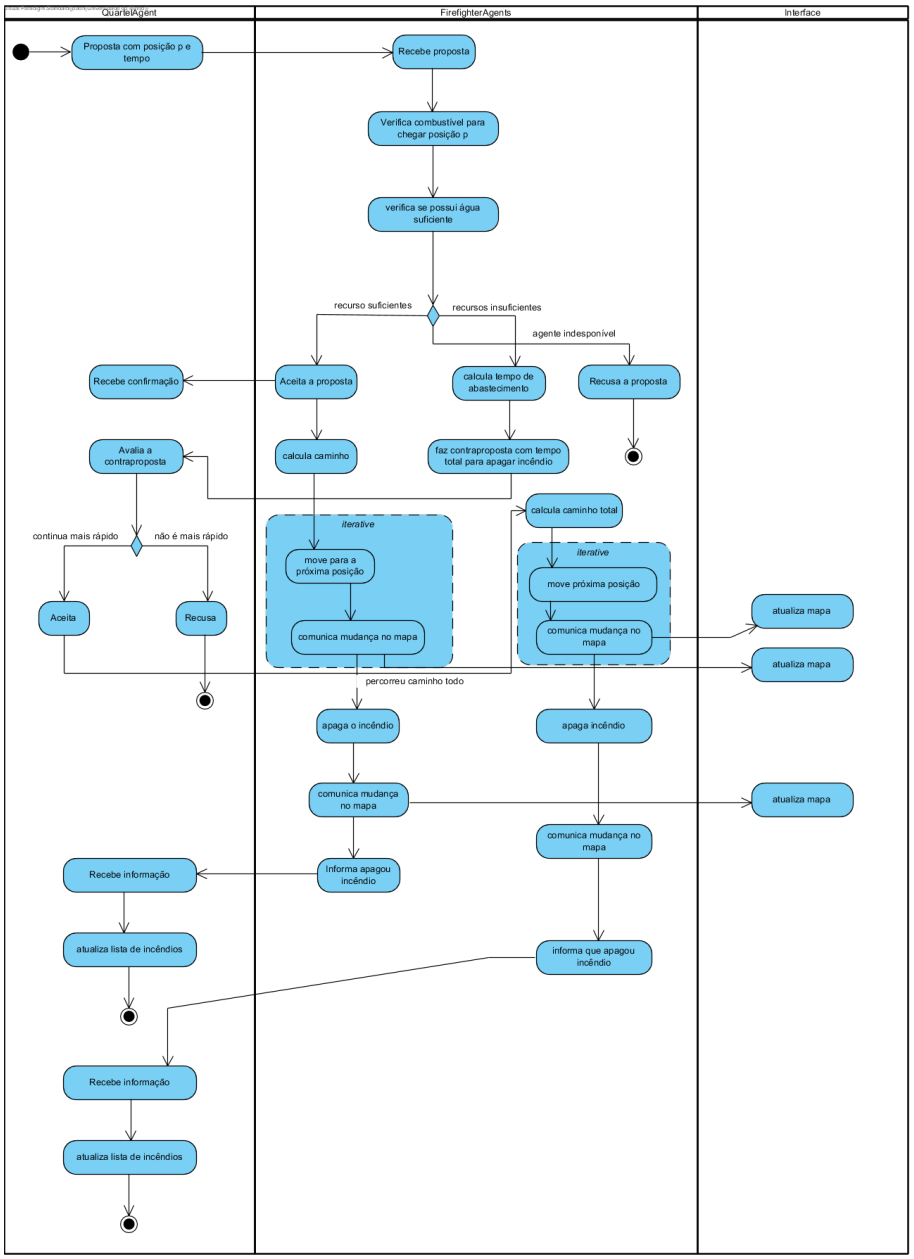

Figura 6.4: Diagrama de atividade para agente apagar incêndio.

7 Máquina de estado

A máquina de estados possui o intuito de identificar os estados diferentes em que o sistema poderá se encontrar em resposta aos eventos que irão acontecer. Como tal, é útil elaborar este tipo de diagramas de modo modelar o comportamento do sistema como um todo.

Como tal, e devido à sua maior complexidade relativamente aos outros, será apresentado de seguida o máquina de estados do apagar incêndio que apresenta todos os diferentes estados relacionado com esta funcionalidade do sistema.

s

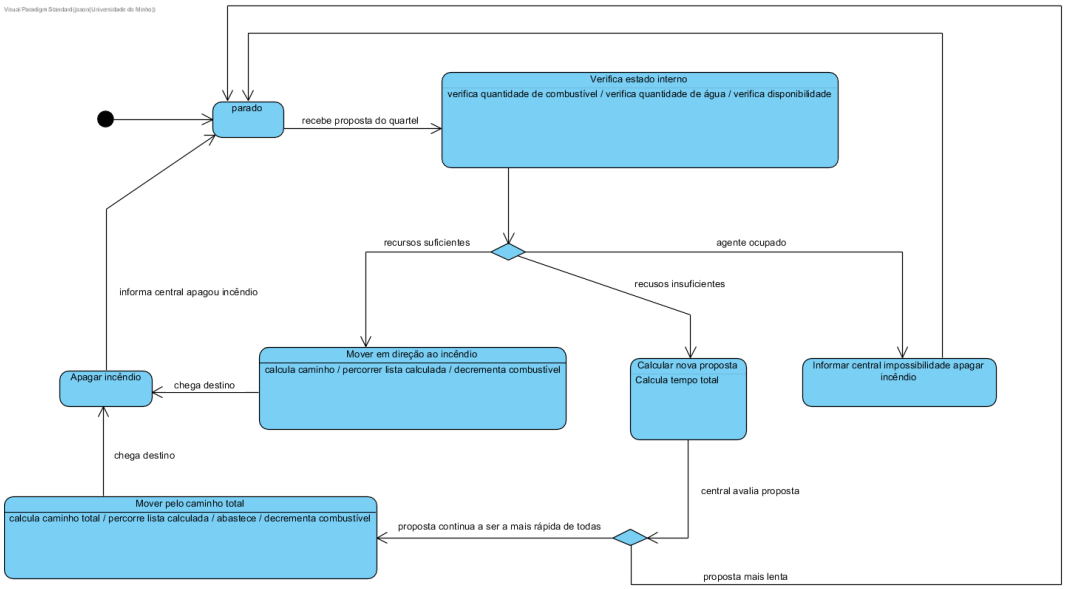

Figura 7.1: Diagrama de estado do apagar incêndio.

Portanto o agente firefighter inicialmente irá se encontrar num estado *idle* sendo que apenas sairá deste estado se receber alguma proposta para apagar um fogo vinda do quartel. Após receber esta proposta, o agente entrar num novo estado onde irá avaliar o seu estado interno verificando o seu nível de combustível, de água e irá também verificar a sua disponibilidade para aceitar esta *proposal*.

De seguida poderá transitar para três estados diferentes consoante a verificação anterior, caso possua recursos suficientes para chegar ao destino e apagar o incêndio irá passar para um estado de movimento em direção ao incêndio onde irá percorrer o caminho que tem a percorrer decrementando também o combustível, de seguida irá entrar no estado de apagar o incêndio e por fim voltará a ficar *idle*.

No caso de possuir recursos insuficientes, irá ter de calcular uma nova contraproposta para oferecer ao agente central, tendo de calcular então o tempo total que corresponde ao tempo de

chegar ao local de abastecimento mais o tempo de deslocação para o incêndio. Esta proposta é enviada para o agente quartel e transita-se para uma estado de verificação desta proposta. Caso esta proposta mesma assim continue a ser mais rápida, entra-se num estado de movimento semelhante ao anterior, uma vez no destino entra-se no estado de apagar o incêndio e novamente o agente fica *idle* no final disto tudo. Caso a proposta seja mais lenta, é necessário informar a central desta ocorrência sendo que o agente irá entrar para um estado *idle* entretanto.

Caso o agente se encontre ocupado com outra delegação para apagar um incêndio, novamente o agente entrará num estado de informar a central da sua impossibilidade de apagar o incêndio, transitando depois para um estado de imobilidade.

8 Conclusão e trabalho futuro

Nesta fase de trabalho foi necessário analisar os requisitos do sistema, e como tal foi necessário analisar o problema proposto de modo a ter uma noção de como funciona o sistema, só depois é que foi possível construir os principais diagramas UML.

Numa primeira instância foi criado o modelo de domínio onde se identificaram as principais entidades que interagem com o sistema, de seguida foi criado o diagrama de classes onde se identificaram as principais classes do eventual sistema desenvolvido bem como alguns métodos, atributos, *behaviours* e interações entre as classes.

De seguida foram abordados os diagramas de sequência onde está explícito de uma forma mais detalhada a comunicação por detrás de alguns dos principais métodos e funcionalidades que o sistema deverá possuir bem como a comunicação entre os agentes, de acordo com o diagrama de classes estabelecido anteriormente.

Finda os diagramas de sequência foram implementados os diagramas de ativadade, sendo estes baseados nos diagramas de sequência criados anteriormente contudo de uma maneira mais compreensível. O último tipo de diagrama *UML* definido pelo grupo foi a máquina de estados que permitiu modelar o comportamento do sistema como um todo.

Numa próxima fase o grupo espera cumprir tudo a que se propôs, e, se possível, no caso de haver tempo, adicionar algumas melhorias ao trabalho. Sem qualquer compromisso, gostaríamos de adicionar a gravidade dos incêndios a partir da proximidade destes com habitações, a propagação de incêndios consoante o vento que se faz sentir, e camiões a não passarem por obstáculos, tendo por isso uma rota mais complicada.

Tendo estabelecido toda esta base por detrás da eventual implementação do sistema, é expectável que esta fase corra com mais facilidade uma vez que o grupo de trabalho possui agora as bases necessárias e bem estabelecidas de modo a implementar o sistema com menos problemas, pelo que também se adivinha então que seja possível cumprir com a data de entrega proposta por todas as razões anteriores.

Bibliografia

[1] Harbers, Maaike, Meyer, John-Jules and van den Bosch, Karel , (2010), Explaining Simulations Through Self Explaining Agents, Journal of Artificial Societies and Social Simulation, 13, issue 1

[2] Moura, Daniel Oliveira, Eugénio. (2006). Fighting Fire with Agents.. CEUR Workshop Proceedings. 223.

[3] Sarmento, Luís et al. “Fighting fire with fear.” (2004).

[4] Moura, Daniel C.. “Coordinating a team of agents in the forest firefighting domain.” (2006).

[5] Abreu, Pedro H. and Pedro Mendes. “Strategies for Route Planning on Catastrophe Environments - Coordinating Agents on a Fire Fighting Scenario.” ICEIS (2009).

[6] F. I. Hernandez, C. A. Canesin, R. Zamora and A. K. Srivastava, "Active power management in multiple microgrids using a multi-agent system with JADE,"2014 11th IEEE/IAS International Conference on Industry Applications, Juiz de Fora, 2014.

[7] Valente, Simão Cardoso. "Implementação de um Sistema Multi-Agente para um Jogo Estratégico de Palavras."(2005).