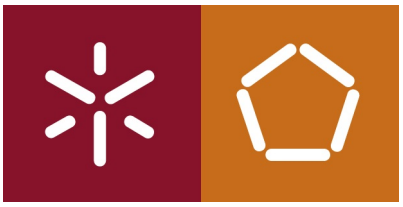

Universidade do Minho
mestrado integrado em engenharia informática

Agentes Inteligentes

Sistemas Inteligentes

(1º Semestre / 4º Ano)

TP1 - Agentes e Sistemas Multiagente

Diogo Braga (a82547)
Pedro Ferreira (a81135)
Ricardo Caçador (a81064)

Conteúdo

1 Introdução

O presente relatório é referente à primeira fase do trabalho prático da unidade curricular de Agentes Inteligentes, integrada no perfil de especialização em Sistemas Inteligentes. Este projeto tem como principal objetivo a conceção e desenvolvimento de um sistema multiagente (SMA) para a monitorização e resolução de catástrofes naturais, através da aplicação de vários sensores virtuais de captura de localização GPS. Neste caso, as catástrofes naturais serão incêndios florestais e o sistema multiagente será composto por diversos tipos de agentes, tais como: os que representam os meios disponíveis para combate aos incêndios, um agente central que representa o quartel de bombeiros e, ainda, um agente incendiário responsável por simular os incêndios.

Nesta primeira fase, era pretendido uma conceção inicial do sistema que incluísse a sua arquitetura, os protocolos de comunicação entre os agentes e uma proposta de resolução inicial. Os protocolos de comunicação deveriam ser especificados recorrendo a `Agent UML`. Pretendia-se ainda um estudo do estado da arte sobre agentes e as suas aplicações a diferentes domínios.

1.1 Estrutura do documento

O documento encontra-se organizado da forma descrita de seguida. Após a introdução, na secção 2 será feita uma contextualização sobre o projeto e apresentado o estado da arte não só sobre agentes e sistemas multiagentes, mas também sobre estratégias de combate a catástrofes naturais. De seguida, na secção 3, serão detalhados os diversos agentes que integrarão o SMA a ser concebido. Na secção 4 é, então, apresentada a proposta de resolução inicial do problema e os diversos protocolos delineados. Por fim, conclui-se apresentando a perspetiva de trabalho futuro.

2 Contextualização e estado da arte

Nos últimos anos, devido a aspetos como o aumento do aquecimento global e a poluição em geral, tem-se verificado um aumento na frequência de ocorrência de desastres naturais. Estes acarretam sérios desafios para o desenvolvimento sustentável da sociedade, pondo muitas vezes em causa a estabilidade sócio-económica das zonas afetadas. Posto isto, de forma a minimizar o impacto destes desastres, torna-se necessário desenvolver métodos de decisão para situações de emergências (EDM - *emergency decision making*), que procuram melhorar a capacidade de resposta. Os incêndios, tema abordado no projeto, são um exemplo de desastre natural.

De forma geral, EDM para desastres naturais engloba todo o tipo de medidas que visam mitigar os efeitos causados por estes, incluindo previsões, monitorização, deteção precoce e planeamento de emergências, alocação de recursos e captação de informação durante o desastre e, por fim, planos de recuperação pós-desastre. Deste modo, e tendo em conta a evolução típica de um desastre natural, podemos identificar quatro momentos distintos no combate aos desastres:

- **Mitigação** - prevenção de desastres através de, por exemplo, monitorização em tempo real;
- **Preparação** - identificação de desastres o mais cedo possível;
- **Combate** - determinação de prioridades durante a ocorrência do desastre e alocação de recursos para combate;
- **Recuperação** - desenvolvimento de um plano de recuperação.

Cada fase tem objetivos e metodologias distintas das restantes. Na figura 1, estas encontram-se representadas com maior detalhe, destacando-se em cada etapa qual o seu objetivo principal e as atividades associadas. A zona que se encontra demarcada a verde corresponde às caraterísticas que serão incorporadas no SMA a desenvolver no âmbito deste projecto.

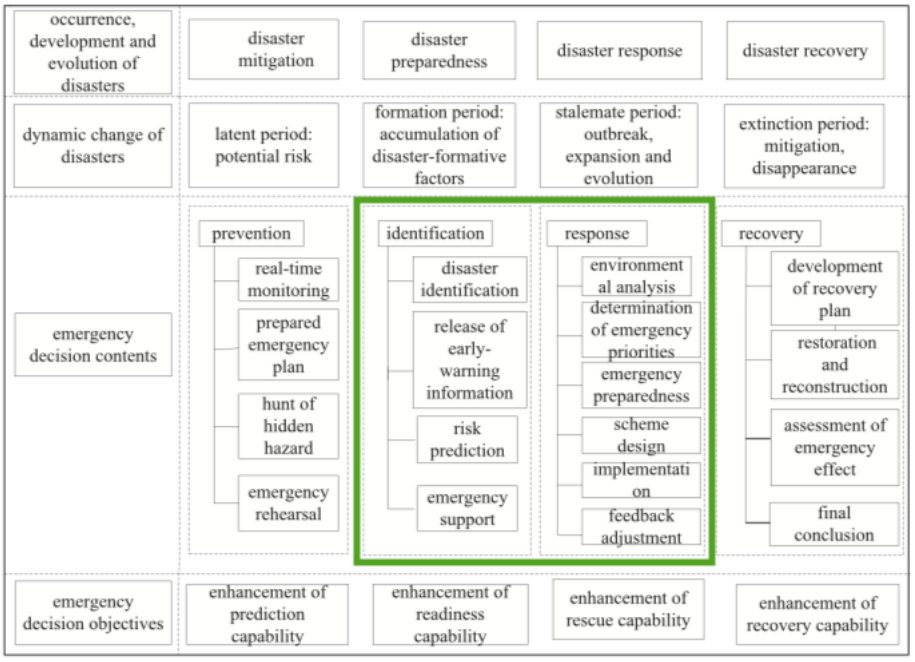

Figura 1: Processo geral de EDMs para desastres naturais [1]

Segundo [1], tem-se registado um aumento crescente do interesse em EDM, sendo que das etapas enunciadas as que recebem mais atenção por parte da comunidade científica são a de resposta e a de preparação. Por outro lado, a fase de recuperação é a menos explorada. No que diz respeito à metodologia seguida nos modelos propostos, os modelos matemáticos são que têm mais representatividade, seguidos de metodologias de simulação e de gestão de conhecimento. Em relação aos sistemas de suporte, os mais usados são, por ordem, sistemas de informação geográfica (SIG), agentes, racionínio baseado em casos (*"case-based reasoning"*), e, mais recentemente, *machine learning* aliado a processamento de linguagens naturais.

A metodologia tem um papel bastante importante no momento de decisão, pois a partir desta é realizada uma evolução da situação. Os métodos de tomada de decisão podem ser classificados nas seguintes categorias:

- **Métodos baseados em Modelos matemáticos**: Abordagem baseada em teoria clássica de decisão e otimização. O problema é encarado como alocação de recursos e análise matemática para gestão de riscos. A título exemplificativo, as *Bayesan networks*, devido à sua caraterística probabilística conseguem lidar com a incerteza subjacente à natureza do problema, tendo contribuído para avaliação de riscos e sistemas de apoio a terramotos e cheias. O processo de decisão de Markov também é aplicado devido à pluralidade de entidades envolvidas no combate aos desastres;

- **Métodos baseados na Evolução da Situação**: Baseiam-se na acumulação de conhecimento e experiência para proceder a uma simulação futura;

- **Métodos baseados em Gestão de Conhecimento**: Existe muita investigação em torno da aplicação de gestão de conhecimento a problemas de monitorização, análise de dados e alocação de recursos, concentrando-se esta na implementação e organização da gestão de conhecimento para EDM.

No processo de EDM, é importante assegurar um tempo de resposta mínimo, sendo, para tal, necessário captar informação de forma pertinente e projeta-la de forma intuitiva e visual para assistir e acelerar a tomada de decisão. Tal é realizado através dos sistemas de suporte à decisão (DSS - *decision support system*). Estes sistemas apresentam diversos dados recolhidos e procedem à modelação da situação real, constituindo, desta forma, um excelente recurso para assistência à tomada de decisão. De entre os diferentes tipos de DSSs existentes, podemos distinguir entre aqueles que se baseiam em:

- **Sistemas de informação geográfica**: Sistemas desenhados para armazenar, gerir e apresentar dados geográficos e espaciais, sendo principalmente utilizados em DSSs direcionados a sismos e cheias;

- **Agentes**: Tema abordado no projeto. Estes são capazes de cooperar de forma a criar organizações multiagente e formar um sistema capaz de implementar uma decisão em caso de emergência. Um exemplo possível é a conjugação de um agente de deteção de problemas, de um agente de gestão de reservatórios e um agente de proteção civil. Estes sistemas são baseados em Inteligência Artificial, e comunicam entre si. Desta forma, os utilizadores humanos não necessitam de participar na tomada de decisão;

- **Racionínio baseado em casos**: Através da acumulação de conhecimento relativo a casos passado, pode ser feito um DSS baseado em CBR, onde se procura realizar a tomada de decisões tendo por base decisões já tomadas no passado;

- **Machine learning aliado a processamento de linguagens naturais**: Mais recentemente, têm surgido sistemas que procuram extrair conteúdo relevante de publicações realizadas em redes sociais, aquando do acontecimento de um desastre natural. Estes partem do pressuposto de que quando um desastre ocorre, grandes quantidades de dados são gerados na em redes sociais, podendo-se, a partir destes, extrair e organizar conhecimento útil para a tomada de decisão.

2.1 Agentes e Sistemas multiagentes

O presente projeto consiste em desenvolver um sistema de EDM, focando-se apenas em incêndios. Tendo por base os tipo de sistemas enumerados anteriormente, podemos concluir que se tratará de um DSS baseado em agentes.

Um agente pode ser visto como algo que corporiza um sistema computacional capaz de uma ação flexível e autónoma, desenvolvido num determinado meio ou sobre um dado universo de discurso [2]. Entre as suas caraterísticas encontram-se aspetos como a proatividade, dinamismo, autonomia, o facto de serem orientados a objetivos e terem perceção sobre o ambiente em que inserem. Um sistema multiagente pode ser definido como uma coleção de agentes autónomos que comunicam entre si para coordenarem as suas atividades, de forma a que, coletivamente, consigam resolver um problema que sozinhos não conseguiriam. As soluções baseadas em agentes vêm, então, substituir a visão de controlo centralizado, rígido e monolítico por um paradigma onde das interações entre individuos surge a caraterística inteligente do sistema.

Segundo [3], os SMA têm vindo a afirmar-se como uma áreas do conhecimento em que se realiza investigação de qualidade, surgindo a todo o momento novos produtos, em diversos setores de atividade económica. Além disso, os SMA encontram-se alinhados com a tendência crescente em desenvolver ferramentas de forma modular, com controlo descentralizado, capazes de responder a disturbios, reconfigurando-se sem necessidade de parar todo o sistema. De seguida, apresentam-se alguns exemplos de aplicações industriais de SMA de forma a melhor caraterizar o estado da arte desta tecnologia:

- Em [4] é apresentado um caso de estudo sobre a implementação de um SMA que servirá como assistente pessoal de viagem. Neste sistema existe um agente responsável por registar as preferências e requisitos do utilizador, produzindo como resultado um conjunto de viagens. Para tal, este agente comunica com outros agentes como o agente de Aviação, Hotel e Aluguer de Carro. Desta forma, é capaz de apresentar uma lista diversificada de opções ao seu utilizador.

- Em [5] são apresentados três exemplos de aplicação de SMA na indústria italiana. O primeiro caso consiste num filtro de notícias de jornais online para apresentação de conteúdo personalizado aos utilizadores. O segundo caso enquadra-se na vigilância marítima, procurando detetar barcos não autorizados. Por fim, o útlimo caso detalha a aplicação de um SMA a um *call center*.

- Em [6] e [7] são explicitadas as vantagens da aplicação de SMAs em contexto de indústria de energia, assim como apresentadas várias recomendações sobre como os desenhar e implementar.

Analisando aplicações existentes de SMAs ao problema abordado neste trabalho, também se podem encontrar várias referências. A título exemplificativo, em [8] é descrita a organização de um SMA desenvolvido com objetivo de ajudar na tomada de decisão em caso de cheias de um rio. Os agentes recebem dados como o nível da água e precipitação, provenientes de diversos sensores, e analisam-nos, de forma a tentar antever o acontecimento de cheias e recomendar medidas

preventivas. Em [9] é apresentada a conceção de um DSS baseado em agentes para gestão de catástrofes em ambientes de múltiplos riscos, isto é, onde é provável que ocorram vários desastres naturais de forma sucessiva.

3 Agentes a desenvolver

Após análise e contextualização sobre o problema proposto, procedemos à apresentação dos agentes que integrarão o SMA que se pretende conceber. Os agentes que constituirão o sistema são os seguintes:

- Agente Incendiário;
- Agente Central;
- Agente Participativo;
- Agente Interface.

De seguida iremos descrever os agentes aqui enumerados, detalhando, para cada um, as caraterísticas identificadas tendo em vista a criação de um sistema o mais completo possível.

3.1 Agente Incendiário

O agente incendiário tem como função iniciar um incêndio na posição em que se encontra no mapa. Após realizar esta ação, deve comunicar ao agente central a ocorrência desse incêndio.

Posto isto, o único dado necessário para representar este agente será a sua posição no mapa. Esta é caracterizada por dois valores que representam as duas coordenadas dum mapa 2D.

3.2 Agente Central (Quartel)

Agente que, tendo conhecimento do estado do mapa, toma decisões com a intenção de resolver os incêndios que surgem. Devido a tal, é ele o responsável pela coordenação e distribuição de tarefas entre os agentes participativos.

Para poder classificar corretamente o grau de perigo de cada um dos incêndios e delegar funções de uma forma correta, o agente central tem de possuir conhecimento do mapa onde decorre a simulação dos fogos. Este inclui diversos tipos de posições, entre as quais:

- Habitações;
- Florestas;
- Postos de abastecimento de água;
- Postos de abastecimento de combustível;
- Zonas de incêndio;

Com base no que foi dito, para além do mapa, o agente central deve ter conhecimento dos meios existentes para combate aos incêndios (agentes participativos) e possuir uma representação de todos os incêndios que se encontram, num determinado momento, ativos, de forma a proceder a uma monitorização e combate adequado. Além disso, consideramos que este terá uma posição no mapa.

3.3 Agentes Participativos

Conjunto de agentes responsáveis pelo processo de combate e extinção de incêndios. Estes encontram-se em constante comunicação com o Quartel de forma a receber informações do estado do mapa e indicações de ações que devem realizar. Existem 3 tipos de agentes, cada um com caraterísticas distintas, tornando-os adequados a certos tipos de incêndios. São eles:

- Drones;
- Aeronaves;
- Camiões;

É ainda importante referir que estes agentes devem ter conhecimento do mapa, sendo este principalmente consultado para averiguação das posições dos pontos de abastecimento de combustível e de água.

3.3.1 Caraterísticas

- **Tipo de agente**: Caraterística que diferencia se o agente participativo é um drone, uma aeronave ou um camião.
- **Posição atual**: Posição no mapa onde se encontra presente o agente participativo.
  - Coordenada x;
  - Coordenada y;
- **Velocidade**: Caraterística que representa a velocidade de movimento do agente no mapa.
  - Drone: 4x
  - Aeronave: 2x
  - Camião: 1x
- **Capacidade máxima de água**: Caraterística que define a capacidade máxima de água do agente.
  - Drone: 2w
  - Aeronave: 15w
  - Camião: 10w
- **Capacidade máxima de combustível**:
  - Drone: 5c
  - Aeronave: 20c
  - Camião: 10c
- **Capacidade atual de água**: Caraterística que define a quantidade atual de água do agente.
- **Capacidade atual de combustível**: Caraterística que define a quantidade atual de combustível do agente.
- **Estado**: Caraterística que define o estado atual do agente, podende este estar livre, em movimentação até ao local de abastecimento ou ocupado a combater um incêndio.

3.4 Agente Interface

O agente interface é o agente que comunica com o agente central para obter o estado atual do mapa e dos agentes, e consequentemente apresentar os resultados da simulação. Este será responsável por apresentar um conjunto de estatísticas ao utilizador, durante o decorrer da simulação. A título exemplificativo, estas estatísticas podem ser representativas do número de fogos ativos num determinado momento, número de unidades de combate a incêndio alocadas e livres, percentagem de mapa ardido, entre outros. O seu objetivo passa por captar métricas que permitam avaliar a perfomance do sistema e a sua capacidade de resolução dos fogos.

4 Proposta de Resolução Inicial

Depois de apresentadas as caraterísticas dos diversos agentes, é importante definir a forma como estes vão interagir de modo a criar um sistema distribuído e dinâmico. Neste capítulo são abordadas várias vertentes com influência nessa área.

4.1 Aspetos a considerar

Antes de abordar secções mais concretas como a arquitetura do sistema e os protocolos de comunicação dos agentes, é importante referir algumas considerações que foram tomadas para desenvolvimento do projeto. Estas considerações procuram extrapolar o enunciado, e delinear certas caraterísticas que gostaríamos de ver implementadas no projeto final. Posto isto, são aqui realçadas para facilitar o entendimento de decisões tomadas nas secções seguintes.

De forma a dotar o SMA a desenvolver com algum realismo, os fogos criados pelo agente incendiário serão dinâmicos, isto é, terão uma evolução temporal, alastrando-se a novas posições do mapa que ainda não tenham sido afetadas. Para tal, o Agente Incendiário terá de conseguir processar a evolução do incêndio, devendo continuar a alertar o quartel sobre a expansão do mesmo.

Com o objetivo de explorar melhor as caraterísticas de cada um dos tipos de agentes participativos supramencionados, serão considerados 2 tipos de células no mapa com influência direta na alocação de recursos a um incêndio:

- Habitações - Zonas de alta prioridade. Serão encaradas como zonas onde existe vida humana pelo que o objetivo do algoritmo de alocação de recursos deve evitar que cheguem incêndios a células deste tipo.
- Zonas de Floresta - De forma a promover estratégias de reação rápida, serão consideradas células representativas de florestas. A sua principal caraterística será a sua atuação como acelaradores de incêndios, isto é, a taxa de propagação do fogo será maior nesta células.

Com a finalidade de extinguir os incêndios, a distribuição de tarefas, por parte do agente central, será feita num só sentido, isto é, não haverá negociação entre o quartel e os meios de combate, limitando-se o primeiro, que possui conhecimento do estado atual dos segundos, a indicar que ações devem ser realizadas e por quem. No entanto, para acrescentar alguma autonomia aos agentes participativos, será permitido que estes se movimentem pelo mapa quando não lhes for atribuída nenhuma tarefa, para reabastecer os tanques de água e combustível. Assim, podemos identificar três estados diferentes no qual um agente participativo se pode encontrar: combatendo o fogo, livre e movendo-se até ao posto de abastecimento. Além disso, este deve informar o quartel sempre que ocorra uma mudança no seu estado e recursos atuais, de forma a que o quartel mantenha uma representação atualizada e coerente sobre o estado dos meios de combate, trocando o mínimo de mensagens necessárias.

4.2 Arquitetura do Sistema

Nesta secção é apresentada a arquitetura do sistema, através do diagrama de classes presente na figura 2. No diagrama é possível identificar:

- Estado dos agentes do sistema, que inclui por exemplo a sua posição;
- Especificação do Agente Participativo em Drone, Aeronave ou Camião;
- A classe Mapa, representativa do ambiente no qual os agentes se encontram inseridos e atuam. Este evolui por contribuições do Agente Incendiário, do Agente Central e dos Agentes Participativos e apresenta, ainda, listas de posições que representam:
  - Incêndios;
  - Postos de abastecimento de combustível;
  - Postos de abastecimento de água;
  - Zonas de habitações;
  - Zonas florestais;
- Estado dos incêndios, sendo cada um deles representado por uma lista de posições (células a arder), duração e gravidade.

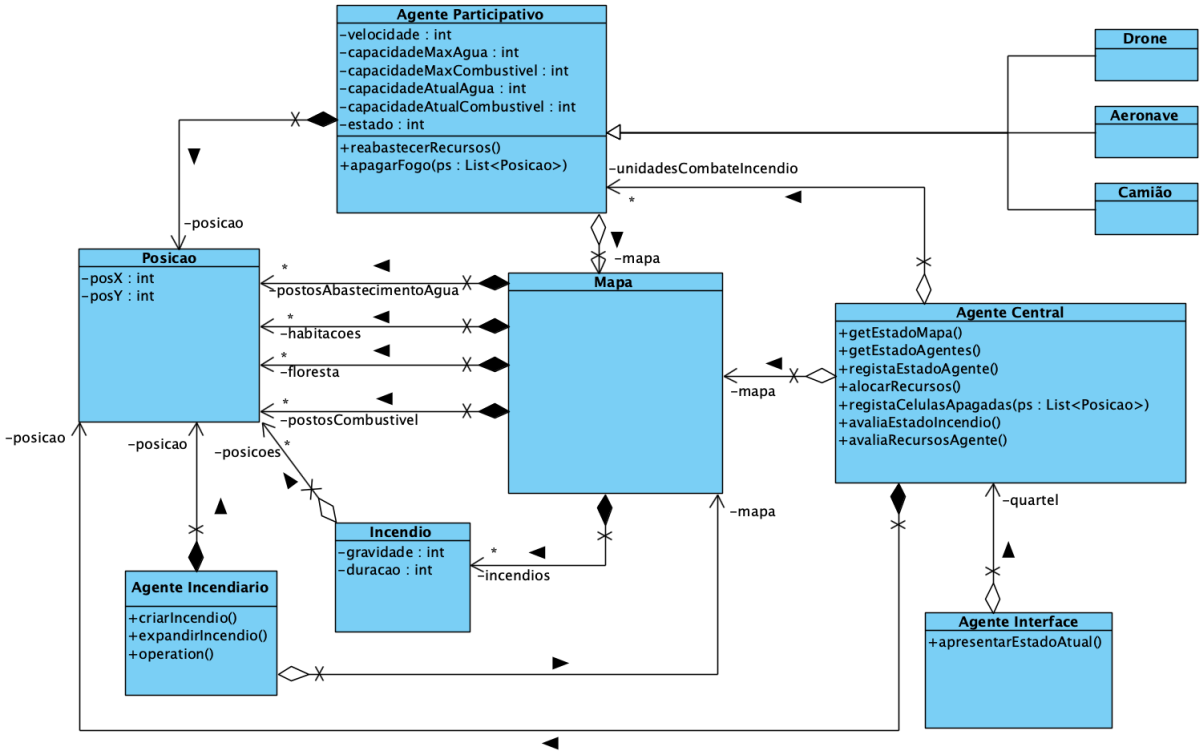

Figura 2: Arquitetura do Sistema

Algumas das relações identificadas serão, na prática, abstraídas pelos mecanismos de comunicação disponibilizados pelo JADE. Será provável que novas relações surjam, por exemplo, o Agente Interface poderá utilizar a classe Mapa, caso seja pertinente para facilitar a amostragem dos dados, sendo um objeto dessa classe enviado pelo Agente Central ao Agente interface aquando da requisição de informação sobre a simulação.

Por fim, também as várias operações identificadas em cada classe, neste caso, agentes, também poderão sofrer alterações. Numa primeira abordagem, estas foram retiradas dos protocolos de comunicação entre agentes que se apresentam neste documento.

4.3 Fluxo de Resolução de Incêndios

De forma a ser mais facilmente compreendido o modo como se processa a resolução de incêndios e a influência dos agentes no sistema, é apresentado na figura 3 o diagrama de atividade que representa a interação entre os agentes, tendo em vista a resolução de um incêndio.

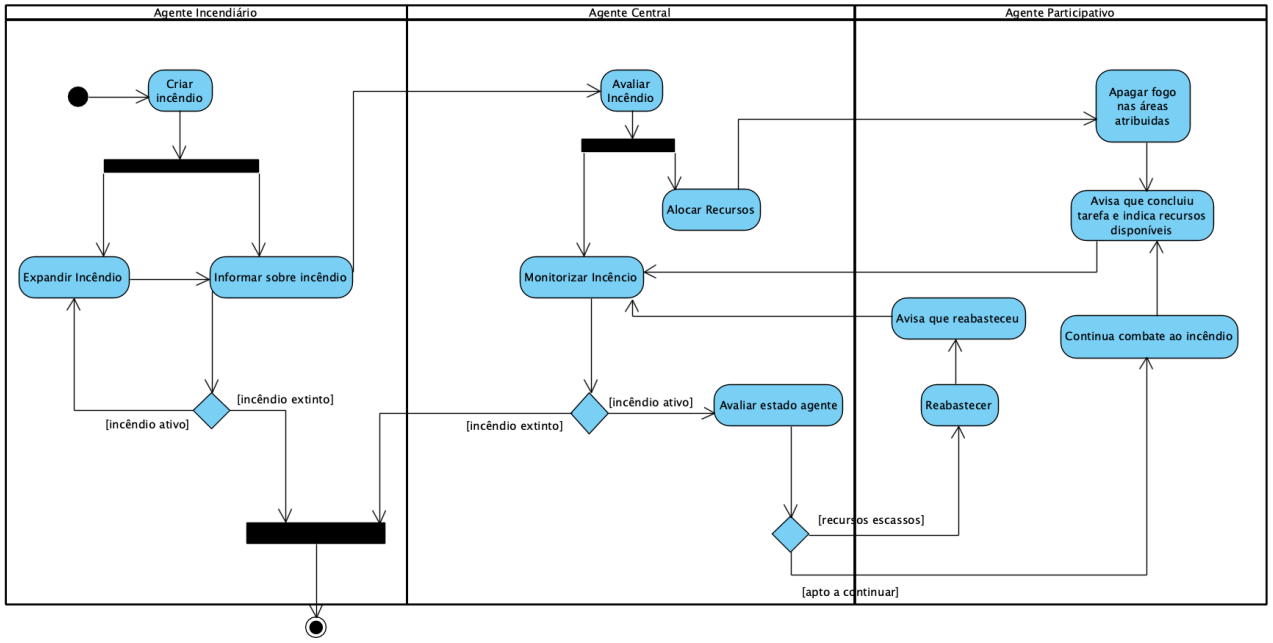

Figura 3: Fluxo de Resolução de Incêndios

Através da observação do diagrama, podemos verificar que o processo tem início com a ignição do fogo, por parte do Agente Incendiário. Este alerta o Agente Central acerca do sucedido e continua a expandir o incêndio, avisando de novas células incendiadas. No lado do Agente Central, as suas ações dividem-se entre monitorizar o incêndio e alocar recursos para o combate. Uma vez delegados os agentes participativos que atuarão no combate e as respetivas tarefas, isto é, células que cada um deve apagar, estes apenas se focam em cumprir a tarefa que lhes for designada. Quando tiverem completado a tarefa, avisam o quartel central que procede a uma avaliação do estado do recursos do agente. Caso o agente possa permanecer em combate, ser-lhe-ão atribuídas novas células para apagar. Caso contrário, deverá ir abastecer a um posto de combustível/água e voltar para a localização do incêndio. O processo termina quando o fogo for extinto.

4.4 Protocolos de Comunicação

De forma a tornar o sistema dinâmico, os agentes apresentados anteriormente na secção 3 têm necessariamente de comunicar entre si. Para que todas estas interações ocorram de forma correta, os agentes cumprem protocolos de comunicação estabelecidos na criação do sistema.

De seguida, apresenta-se o os diversos protocolos de comunicação entre os agentes, introduzindo-se o tema com a definição de um diagrama de colaboração simplificado que serve para identificar as *perfomatives* utilizadas entre cada par de agentes e em que sentido estas são usadas.

O panorama geral de comunicação entre os agentes encontra-se representado na figura 4:

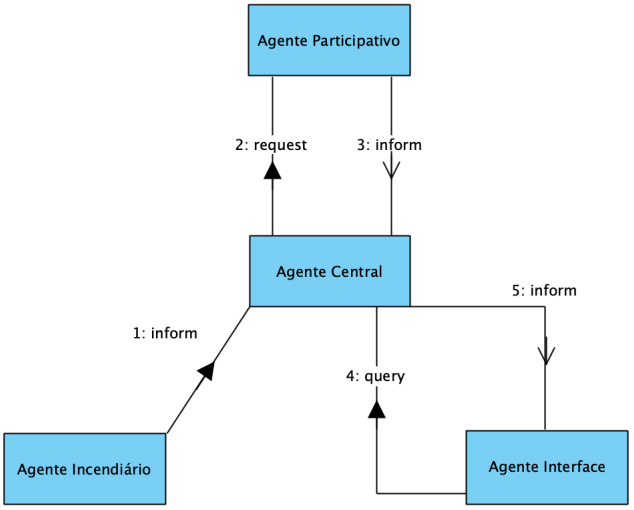

Figura 4: Panorama geral de comunicação entre agentes

Observando a figura, podemos verificar que entre o Agente Incendiário e o Agente Central existem mensagens do tipo `inform`. Estas correspondem aos avisos sobre células onde o Agente Incendiário ateou fogo. As mensagens do tipo `request` entre o quartel e os Agentes Participativos correspondem à delegação de tarefas. Seguindo a mesma linha de raciocínio, as mensagens do tipo `inform` entre Agentes Participativos e o quartel servem para informar sobre a conclusão destas mesmas. Noutra perspetiva, estas podem também corresponder a avisos na alteração do estado do agente. Por fim, é possível constatar o pedido de dados por parte do Agente Interface ao Agente Central, através da mensagem de tipo `query`, e a respetiva resposta, por meio de um mensagem `inform`.

Apresentada uma visão mais genérica sobre o sistema, procedemos então para a definição dos diferentes protocolos de comunicação entre agentes.

4.4.1 Protocolo entre Agente Incendiário e Agente Central

Este protocolo de comunicação, representado na figura 5, estabelece a passagem de informação do agente incendiário para o agente central. A informação transferida é relativa à criação de incêndios, e sucessivas expansões.

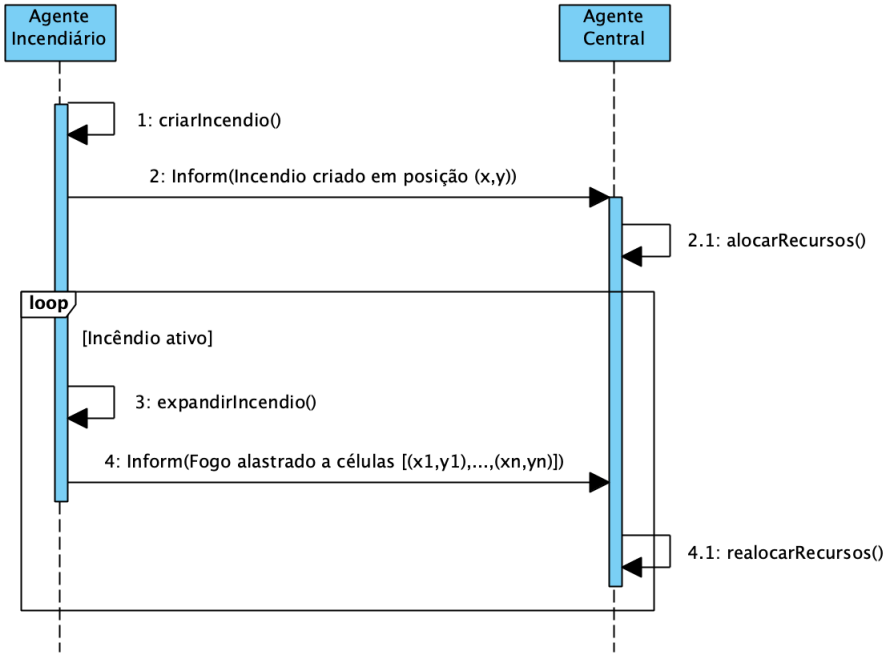

Figura 5: Comunicação Agente Incendiário - Agente Central

Analisando o diagrama, podemos verificar a existência de dois momentos distintos: o de ignição de fogo e o de expansão do mesmo. Por cada conjunto de células a que o agente incendiário alastre o fogo, irá informar o quartel, devendo este reagir a este estímulo, nomeadamente através da delegação de novas entidades para combate ao incêndio.

4.4.2 Protocolo entre Agente Central e Agente Participativo (Estado ocupado)

Este protocolo de comunicação, representado na figura 6, modela a troca de informação entre o agente central e qualquer agente participativo que tenha sido selecionado para combater um determinado incêndio. Assim, referimo-nos ao agente participativo como ocupado porque aquando da execução deste protocolo ele é monitorizado pelo agente central.

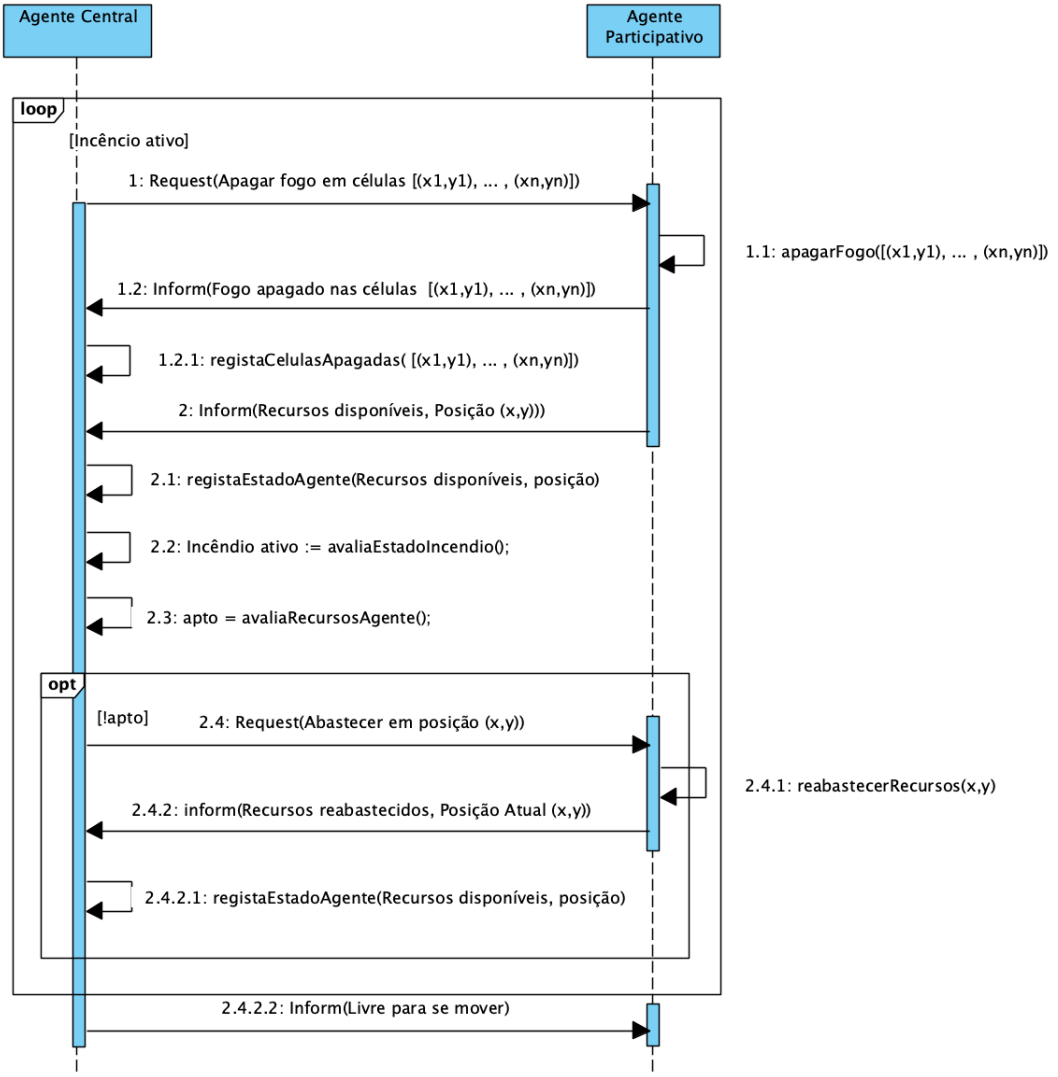

Figura 6: Comunicação Agente Central - Agente Participativo (Ocupado)

Analisando a figura 6, podemos verificar que este protocolo se aplica enquanto o incêndio em causa estiver ativo. O algoritmo seguido é bastante simples e começa pela delegação de um conjunto de células ao agente participativo para que este as apague. Quando a tarefa estiver realizada, deve informar o quartel central, sendo que nesse momento é feita uma revisão do estado do incêndio e dos recursos do agente. Caso o agente participativo não apresente recursos para executar a próxima tarefa, deverá deslocar-se a um ponto de abastecimento. Caso contrário, e a sua presença ainda seja precisa, ser-lhe-á atribuída uma nova tarefa. Este procedimento repete-se até que o incêndio esteja extinto, altura em que o agente central avisa o agente participativo que este se encontra livre para se mover.

4.4.3 Protocolo entre Agente Participativo (Livre) e Agente Central

Este protocolo de comunicação define a passagem de informação entre o agente participativo e o agente central. A informação transferida é relativa aos recursos disponíveis.

Neste caso referimo-nos ao agente participativo como livre porque ele não se encontra alocado a qualquer incêndio e, consequentemente, possui autonomia para reabastecer.

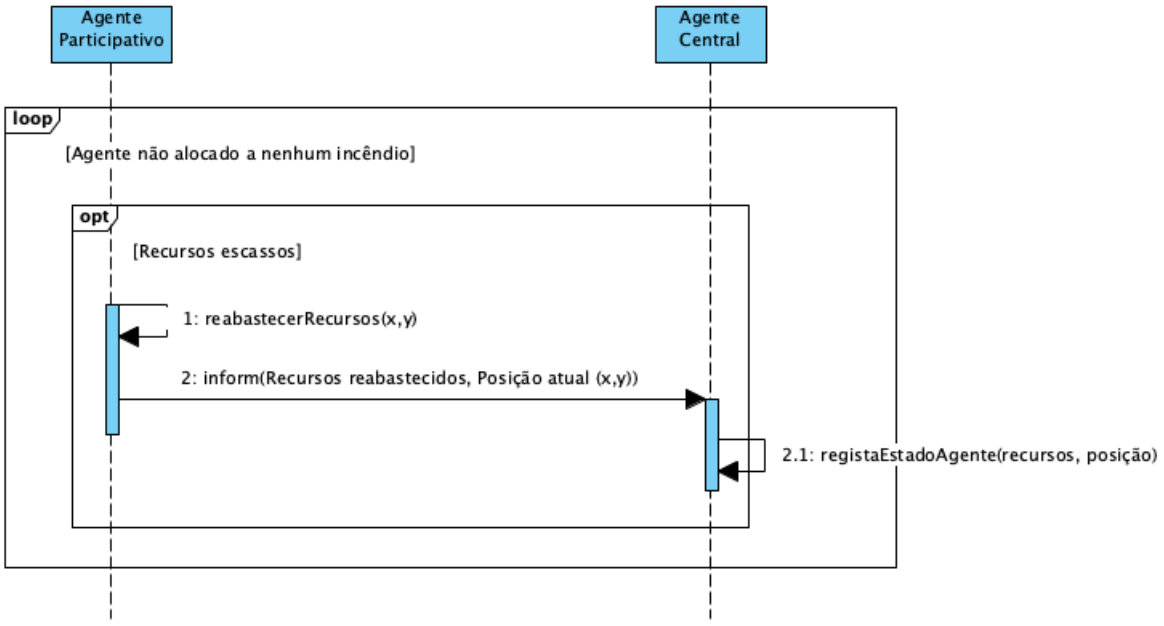

Figura 7: Comunicação Agente Participativo (Livre) - Agente Central

4.4.4 Protocolo entre Agente Interface e Agente Central

Este protocolo de comunicação realiza a requisição de informação por parte do agente interface em relação ao agente central. A informação transferida, em paralelo, é relativa ao estado do mapa e aos agentes participativos.

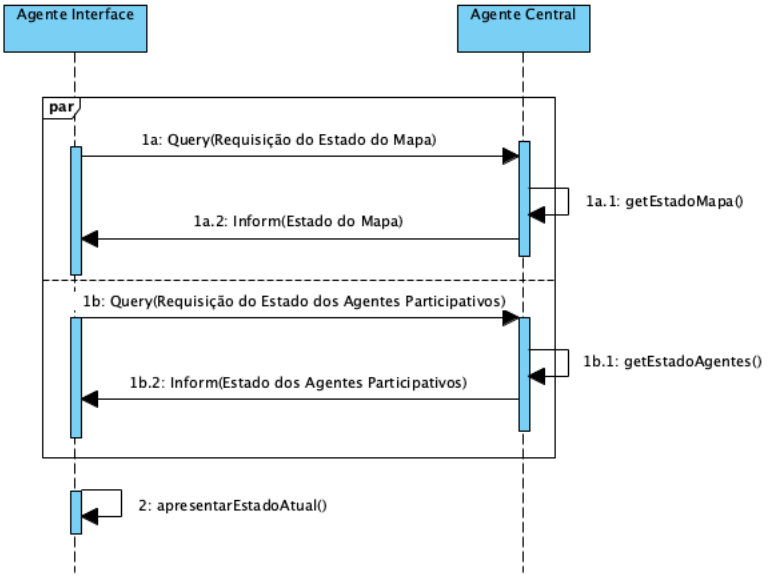

Figura 8: Comunicação Agente Interface - Agente Central

4.5 Estado Interno do Agente Participativo

Uma vez que os agentes participativos são os que apresentam maior dinamismo, podendo encontrar-se em três estados distintos durante a execução da simulação, apresenta-se, de seguida, um diagrama de máquina de estados representativo do seu comportamento interno.

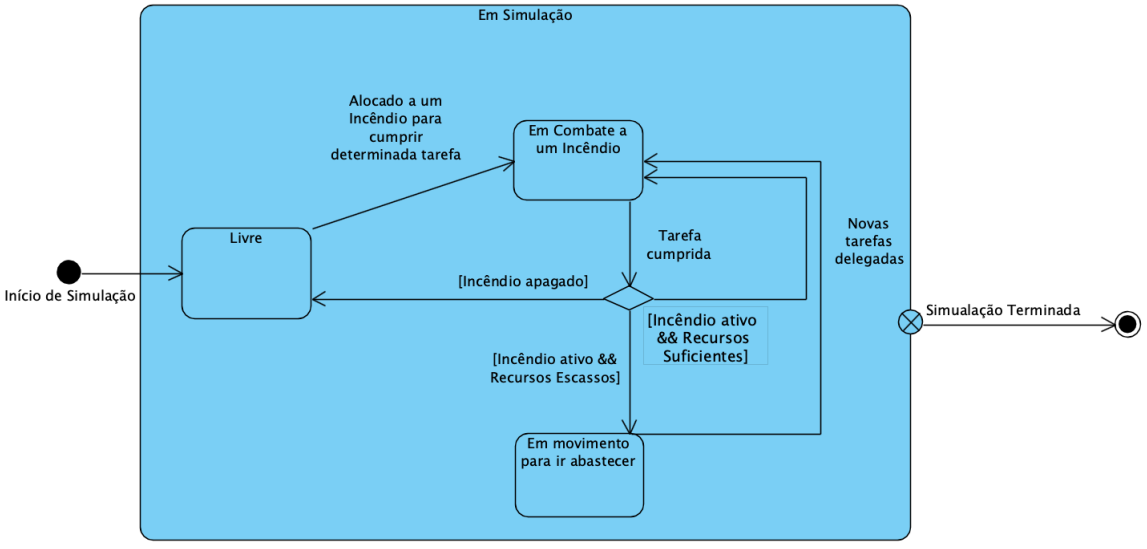

Figura 9: Estado interno dos agentes participativos

Neste diagrama, podemos constatar a existência dos três estados já referidos. Todos os agentes começam como estando livres, isto é, no momento inicial da simulação ainda nenhuma tarefa foi distribuída. Aquando da ocorrência de fogos, os agentes cujos serviços forem requisitados passarão a estar no estado de combate a incêndio. Quando os seus recursos forem escassos, será necessário proceder ao reabastecimento dos mesmo. Estes voltarão a ficar livres quando o incêndio for apagado.

5 Implementação

Nesta secção será apresentada a extratégia seguida na implementacao, acompanhada excertos de código relevantes, onde é possível visualizar os *Behaviours* necessários para o bom funcionamento dos agentes e do sistema, assim como a comunicação estabelecida entre os diversos intervenientes, e as respetivas *performatives*.

5.1 Considerações iniciais

Tendo em vista o desenvolvimento de um sistema que se aproximasse de um simulador, sendo este totalmente controlado por agentes, o diagrama presente na figura 4 sofreu algumas alterações.

Na visão inicial que tínhamos sobre o sistema, as funções delegadas ao Agente Interface apenas passariam por, periodicamente, pedir informação ao Agente Central sobre o estado da simulação, procedendo à sua representação gráfica. Ora, para permitir múltiplas simulações sem ser necessário reiniciar o programa, o Agente Interface deveria ser capaz de permanecer ativo entre simulações. Além disso, os restantes agentes teriam de ser terminados ao finalizar uma simulação, e reiniciados aquando do começo de uma nova.

Assim, o Agente Interface, ao receber indicação para terminar a simulação, passou a ser responsável por indicar aos restantes agentes que deveriam terminar a sua atividade. Para tal, foi necessário adicionar mecanismos de comunicação não representados na figura 4. Nesta nova visão, quando o utilizador pretende terminar a simulação, o Agente Interface envia uma mensagem com *performative* **Request** a todos os restantes agentes. Por sua vez, estes terminam a sua atividade com a receção desta mensagem. Quando o utilizador desejar iniciar uma nova simulação, o Agente Interface executa o método que inicia os restantes agentes.

5.2 Mapa

Existem quatro tipos de objetos fixos distribuídas pelo mapa. A seguir são enunciadas, assim como são explicadas as formas de distribuição das mesmas:

- Habitação: colocada aleatoriamente, mas tendência a realizar grupos para concretizar a ideia de cidade (conjunto de habitações);
- Ponto de floresta: colocado aleatoriamente, mas tendência a realizar grupos para concretizar a ideia de floresta;
- Posto de combustível: distribuídos uniformemente por quadrante, sendo a posição dentro de cada quadrante aleatória;
- Posto de água: colocado aleatoriamente, mas tendência a realizar linhas para concretizar a ideia de rios;

Os agentes participativos, restantes objetos presentes no mapa, são distribuidos aleatoriamente.

5.3 Agente Incendiário

O agente incendiário é responsável por iniciar células de fogo e expandir as mesmas, consoante algumas condições definidas. Os *Behaviours* para isso definidos são apresentados de seguida.

Para além dos *behaviours* associados à criação e expansão de incêndios, este agente possui ainda um ***CyclicBehaviour*** que espera mensagens provenientes do agente interface, do tipo *performative* **Request**, para saber quando deve terminar a sua atividade.

```
protected void setup(){

    this.centralAgent = DFManager.findSingleAgent(this, "Central");
    DFManager.registerAgent(this, "Incendiario");
    ...
    /* Colocar celula de fogo */
    addBehaviour(new PlaceFire(this, this.freqIncendio));

    /* Receber mensagens do Agente Interface */
    addBehaviour(new MsgReceiver());
}
```

PlaceFire

A partir de um ***TickerBehaviour***, com um *tick* definido pelo atributo *freqIncendio*, é selecionada, de forma aleatória, uma célula do mapa para ser incendiada.

O agente comunica com o agente central quais as células incendiadas, através da *performative* **Inform**.

Por cada novo incendio criado, é adicionar um *Behaviour* responsável por expandi-lo (SpreadFire).

```
protected void onTick() {
    ...
    FireAlert fa = new FireAlert(this.fireId, p, ts);

    /* Comunicacao com o Agente Central */
    ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
    msg.addReceiver(central);
    msg.setContentObject(fa);
    send(msg);

    /* Colocar expansao nos fogos ativos */
    addBehaviour(new SpreadFire(this, this.freqExpansao, p, this.fireId));
}
```

SpreadFire

A partir de um ***TickerBehaviour***, com um *tick* definido pelo atributo *freqExpansao*, é selecionada, de forma aleatória, uma célula adjacente às células que o incêndio ocupa no mapa.

Também neste momento, o agente comunica com o agente central quais as novas células incendiadas, através da *performative* **Inform**.

```
protected void onTick() {
    ...
    FireAlert fa = new FireAlert(this.fireId, celulasIncendiadas);

    /* Comunicacao com o Agente Central */
    ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
    msg.addReceiver(central);
    msg.setContentObject(fa);
    send(msg);

}
```

5.4 Agente Central

O agente central é responsável por coordenar a maior parte da comunicação existente no sistema. Recebe os alertas de incêndio do agente incendiário, as alterações de estado de todos os agentes participativos, e do agente interface recebe os pedidos de requisição sobre a informação do estado. Por outro lado, tanto envia tarefas aos agentes participativos, como envia a informação do estado ao agente interface.

```
protected void setup(){
    DFManager.registerAgent(this, "Central");
    ...
    /* Receber mensagens dos agentes */
    addBehaviour(new ReceiveInfo());
}
```

ReceiveInfo

Para receber toda a informação de modo a coordenar o sistema e os agentes intervenientes, o agente possui um ***CyclicBehaviour*** que espera mensagens provenientes de todos eles, de forma a poder responder aos pedidos requeridos.

```
private class ReceiveInfo extends CyclicBehaviour {
        public void action() {
            ACLMessage msg = receive();
            if (msg != null) {

                if (msg do Agente Incendiario) {
                    ...
                    FireAlert fa = msg.getContentObject();
                    processFireAlert(fa);
                }
                else if (msg de um Agente Participativo) {
                    ...
                    AgentStatus st = msg.getContentObject();
                    atualizarEstadoAgente(sender, st);
                }
                else if (msg do Agente Interface) {
                    ...
                    sendSimulationInfo(msg);
```

```
                }
            }
        }
    }
```

Caso a mensagem provenha do Agente Incendiário, são realizados os seguintes passos:

- Registo do incêndio, ou atualização caso se trate duma expansão;
- Atribuição da tarefa a um agente participativo, realizando um algoritmo para encontrar o agente que demora menos tempo a executar a tarefa em causa. O algoritmo de alocação de agentes a um incêndio passa por procurar qual o agente que consegue em menor tempo, deslocar-se até à localização do incêndio. Caso um agente esteja ocupado com outras tarefas, o tempo que demorará a conclui-las será também contabilizado. Para além disso, eventuais necessidade de reabastecer recursos também acrescem ao tempo em que o agente demorará a atingir o incêndio. Assim:
  - Se for necessário abastecer água, encontra o posto de água que seja mais rápido para chegar, em direção ao incêndio a si atribuído;
  - Se for necessário abastecer combustível, encontra o posto de combustível que seja mais rápido para chegar, em direção ao incêndio a si atribuído;
  - Se o agente delegado ao incêndio necessitar de reabastecer tanto água como combustível, então é escolhido o caminho mais curto entre a posição do agente, um posto de água, um posto de combustivel e a posição do incêndio.
- Caso o incêndio aconteça numa floresta (taxa de propagação maior), é também alocado o segundo agente participativo que demora menos tempo a se deslocar ao local, para prevenção de possível expansão.

Ao agente(s) selecionado(s), para além de lhes ser delegada a tarefa de apagar o fogo numa determianda célula, ou deslocar-se de forma preventiva, também podem ser delegadas tarefas relacionadas com o reaprovisionamente de recursos, indicando-lhe onde e o que devem abastecer. Os agentes cumprem as tarefas delegadas pelo agente central de forma sequencial, pela ordem de atribuição imposta pela agente central.

A alocação das tarefas estabelecidas a um agente participativo, pelo algoritmo do agente central, são enviadas através de um ***OneShotBehaviour***, com a *performative* **Request**. O processo é finalizado com o envio das tarefas para o agente escolhido através da classe *AssignTask*.

```
private class AssignTask extends OneShotBehaviour {
    public AssignTask(AID agentAID, Tarefa... tarefas) {
        this.tarefas = new LinkedList<>();
        this.agent = agentAID;

        Arrays.stream(tarefas).forEach(t -> {
            this.tarefas.add(t);
            ...
        });
    }

    public void action() {
        ACLMessage msg = new ACLMessage(ACLMessage.REQUEST);
        msg.addReceiver(agent);
```

```
15          msg.setContentObject(this.tarefas);
16          send(msg);
17      }
18  }
```

Caso o agente central receba uma mensagem de um Agente Participativo, é porque o estado desse agente se alterou e este se encontra a informa o agente central sobre a sua nova situação. É também neste momento de comunicação que o agente central adquire, por exemplo, as informações relativas às tarefas realizadas, às posições dos agentes e às células ardidas, tendo por base o tempo que o agente demorou a executar certa tarefa. Desta forma adquire as informações necessárias para futuramente as enviar para o Agente Interface.

Caso a mensagem provenha do Agente Interface, significa que é uma requisição do estado atual do sistema, e tal é enviado com intenção de ser realizada uma atualização na interface providenciada pelo sistema.

5.5 Agente Participativo

O agente participativo é responsável por apagar os incêndios. Para isso, cada agente abastece, periodicamente, o seu combustível e o seu tanque de água. Todas essas decisões são enviadas, como tarefas, pelo agente central, que controla o estado dos agentes participativos.

Este agente possui ainda a capacidade de, sem tarefas atribuídas, tomar a iniciativa para ir abastecer os seus depósitos.

```
protected void setup(){
    DFManager.registerAgent(this, "Agent");
    this.centralAgent = DFManager.findSingleAgent(this, "Central");
    ...
    this.tbf = new ThreadedBehaviourFactory();

    /* Receber tarefas do Agente Central */
    this.addBehaviour(tbf.wrap(new TaskReceiver()));
}
```

TaskReceiver

Para receber a informação sobre a tarefa a si alocada, o agente possui um ***CyclicBehaviour*** que espera mensagens provenientes do agente central. Caso este se encontre disponível para realizar a tarefa no momento, cria um ***OneShotBehaviour*** para a execução da mesma. Caso contrário, a tarefa fica armazenada numa coleção de tarefas agendadas para execução posterior.

```
class TaskReceiver extends CyclicBehaviour {
    public void action() {
        ACLMessage msg = receive();
        if (msg != null) {
            if (msg do Agente Central) {
                tarefas = msg.getContentObject();
                ...
                for(Tarefa t : tarefas) {
                    tarefasAgendadas.add(t);
                }
                if(disponivel==true){
```

```
                    /* Realizar tarefa */
                    addBehaviour(tbf.wrap(new PerformTasks()));
                }
            }
        }
    }
}
```

PerformTasks

Este *Behaviour* do agente participativo é responsável pela execução da tarefa, incluindo a movimentação até ao destino requerido, e a finalidade em si, ou seja, o ato de apagar o fogo, encher algum ou ambos os depósitos, ou simplesmente realizar prevenção.

No final, caso o agente não possua mais nenhuma tarefa agendada, toma a iniciativa para ir reabastecer os seus tanques, e tal é realizado na classe *RefillTanks* através dum ***OneShotBehaviour***.

```
class PerformTasks extends OneShotBehaviour {
   public void action(){
       while(freeModeActive);

       while(tarefasAgendadas.peek() != null) {
           ...
           Tarefa t = tarefasAgendadas.poll();

           boolean deslocacaoCompleta = moveToPosition(t.posicao, t, false
               );

           if(t.tipo == Tarefa.APAGAR)
               apagarFogo(t);
           else if(t.tipo == Tarefa.ABASTECERCOMB)
               abastecerComb();
           else if (t.tipo == Tarefa.ABASTECERAGUA)
               abastecerAgua();
           else if(t.tipo == Tarefa.PREVENIR) {
               if (deslocacaoCompleta)
                   prevenir();
           }
           tarefasRealizadas.add(t);
       }

       if(tarefasAgendadas.size() == 0){
           ...
           rt = new RefillTanks();
           /* Iniciativa para ir abastecer os tanques */
           addBehaviour(tbf.wrap(rt));
       }
   }
}
```

A classe *RefillTanks* possui um algoritmo para indicar ao agente quais os locais mais próximos para reabastecer, e dessa forma, estar prevenido, e consequentemente melhor preparado, para responder aos pedidos do agente central.

5.6 Agente Interface

O agente interface é responsável pela atualização e funcionamento da interface fornecida ao utilizador pelo sistema.

O agente possui um ***CyclicBehaviour*** para receber informação. Este é executado através do método *InfoReceiver* que recebe do agente central, recorrentemente, o estado do sistema. A partir dessa informação, atualiza a *GUI* disponibilizada.

O agente possui, também, um ***TickerBehaviour*** para requisitar informação ao agente central, através da *performative* **Query_Ref**.

InfoReceiver

```
class InfoReceiver extends CyclicBehaviour {
    public void action() {
        ACLMessage msg = receive();
        if (msg != null) {
            ...
            if (msg do Agente Central) {
                DeltaSimulationStatus stats = msg.getContentObject();
                updateGui(stats);
            }
            ...
        }
    }
}
```

6 Interface

Nesta secção será apresentado o resultado final da interface, assim como os métodos desenvolvidos para implementação dos gráficos que decidimos utilizar para analisar as simulações que são efectuadas. Todos os gráficos foram feitos utilizando o **JFreeChart**.

6.1 Interface da Simulação

A interface que permite correr as simulações e visualizar as diferentes interações entre os agentes e o meio em que estes se inserem, encontra-se representada na figura 10. Esta foi desenvolvida com recurso a **Swing** e encontra-se sobre alçada do Agente Interface.

Tendo como objetivo permitir diferentes simulações seguidas, sem que seja necessário reiniciar o programa ou alterar ficheiros de configuração, a interface contempla 3 botões. Enquanto que o primeiro serve para criar um novo mapa, os restantes servem para iniciar/parar a simulação. Além disso, existem 8 caixas de texto que permitem alterar parâmetros que serão utilizados na simulação. Estes dizem respeito não só ao número de agentes participativos, mas também às carateristicas do mapa onde estes atuarão.

Figura 10: Interface da Simulação

6.2 Histograma Simples

O primeiro gráfico apresentado é um histograma simples que apresenta a quantidade de células ardidas por tipo de célula (habitação, ponto florestal, posto de combustível ou célula normal). Nesta classe utilizámos os métodos fornecidos pelo JFreeChart para criar um Histograma Simples de apenas 1 barra por classe. Para ir buscar os valores de contagem de células foram feitas 4 funções idênticas que têm como objetivo contar as células ardidas de um dado tipo.

```
1  public int contaHabArdidas () {
2          int cont = 0;
3          if (this.mapa.areaArdida  !=  null) {
4              for (Posicao p : this.mapa.areaArdida) {
5                  if (this.mapa.hab(p)) cont++;
6              }
7              return cont;
8          }
9          return 0;
10     }
```

Assim na criação do dataset para inserir no histograma é chamada esta função consoante o tipo de células ardidas que queremos contar.

```
1  private CategoryDataset createDataset() {
2
3          var dataset = new DefaultCategoryDataset();
4          dataset.setValue(contaFlorestaArdidas(), "Celulas Ardidas", "Ponto
               Florestal");
5          dataset.setValue(contaHabArdidas(), "Celulas Ardidas", "Habitacao")
               ;
```

```
6           dataset.setValue(contaPostosCombArdidas(), "Celulas Ardidas", "
                Postos Combustiveis");
7           dataset.setValue(contaOutrasArdidas(),"Celulas Ardidas", "Vazias");
8           return dataset;
9       }
```

Na figura seguinte é apresentado um exemplo de um gráfico num simulação para um tamanho de mapa 10:

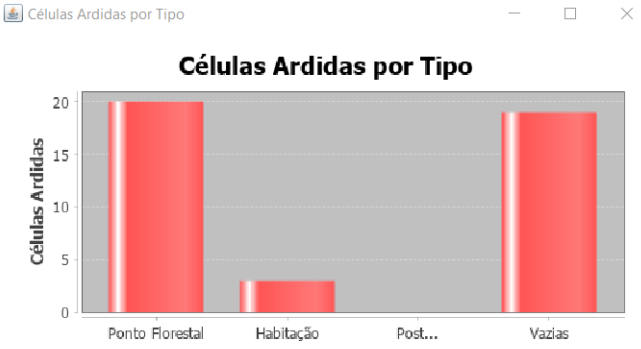

Figura 11: Tipo de Células Ardidas

6.3 Histograma Agrupado de 4 Barras

O segundo gráfico que decidimos implementar foi um gráfico que permitisse analisar as tarefas mais praticadas por cada agente. Assim escolhemos este tipo de histograma que agrupa 4 barras para as 4 diferentes tarefas e define 3 classes para os 3 tipos de agentes.

Utilizando os métodos fornecidos pelo JFreeChart criámos então este histograma e para a criação do seu dataset foi também necessário aplicar uma função que contasse as tarefas por tipo de tarefa e por tipo de agente. Para isso foi necessário passar um Map¡String,List¡Tarefa¿¿ pelo Delta Simulation Status que contém a lista de Tarefa realizadas por cada agente.

```
1   public Map<Integer,Integer> contagemPorAgente (String s) {
2               Map<Integer,Integer> contas = new HashMap<Integer,Integer>();
3               int comb=0;
4               int apagar=0;
5               int prevenir =0;
6               int agua=0;
7               if (this.tarefasRealizadas!= null) {
8               for (Map.Entry<String,List<Tarefa>> entry : this.
                    tarefasRealizadas.entrySet()) {
9                   if ( entry.getKey().equals(s)){
10                      for ( Tarefa t : entry.getValue()) {
11                          if (t.tipo == 1) comb++;
12                          else if (t.tipo == 2) apagar++;
13                          else if (t.tipo == 3) prevenir++;
14                          else if (t.tipo == 4) agua++;
15                      }
16                  }
17              }
```

```
18              }
19              contas.put(1,comb);
20              contas.put(2,apagar);
21              contas.put(3,prevenir);
22              contas.put(4,agua);
23              return contas;
24      }
```

createChart

Função responsável pela criação do Histograma

```
1   private JFreeChart createChart(CategoryDataset dataset) {
2
3           JFreeChart barChart = ChartFactory.createBarChart(
4                   "Quantidade de Tarefas por Tipo de Agente",
5                   "Tarefa",
6                   "Quantidade",
7                   createDataset(),
8                   PlotOrientation.VERTICAL,
9                   true, true, false);
10
11          ChartPanel chartPanel = new ChartPanel( barChart );
12          chartPanel.setPreferredSize(new java.awt.Dimension( 560 , 367 ) );
13          setContentPane( chartPanel );
14
15          return barChart;
16      }
```

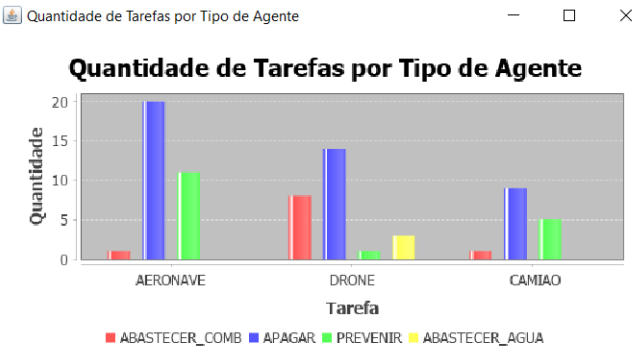

Figura 12: Tarefas por Tipo de Agente

6.4 Gráfico Circular

Por último, decidimos implementar um gráfico que apresentasse uma visão mais geral da simulação e do trabalho realizado pelos agentes e optámos por um gráfico circular que apresentasse uma ideia das tarefas mais realizadas pelos agentes.

Mais uma vez, utilizando o que o JFreeChart já oferece, construímos o gráfico circular, estabelecendo o número de setores e os seus nomes. Para a criação do dataset foi criada uma função (apresentada a seguir) que conta o total de tarefas realizadas por todos os agentes, através

da iteração sobre o Map¡String,List¡Tarefa¿¿ tarefasRealizadas já utilizado no dataset do gráfico anterior.

contagemTotal

```
public Map<Integer,Integer> contagemTotal () {
        Map<Integer,Integer> contas = new HashMap<Integer,Integer>();
        int comb=0;
        int apagar=0;
        int prevenir =0;
        int agua=0;
        if (this.tarefasRealizadas!= null) {
            for (Map.Entry<String,List<Tarefa>> entry : this.
                tarefasRealizadas.entrySet()) {
                    for ( Tarefa t : entry.getValue()) {
                        if (t.tipo == 1) comb++;
                        else if (t.tipo == 2) apagar++;
                        else if (t.tipo == 3) prevenir++;
                        else if (t.tipo == 4) agua++;
                    }

            }
        }
        contas.put(1,comb);
        contas.put(2,apagar);
        contas.put(3,prevenir);
        contas.put(4,agua);
        return contas;
    }
```

Esta função é então utilizada na criação do dataset do gráfico, escolhendo para cada sector, o tipo de tarefa.

createDataset

```
  private  PieDataset createDataset( ) {
        DefaultPieDataset dataset = new DefaultPieDataset( );
        dataset.setValue( "Abastecimento C" , contagemTotal().get(1));
        dataset.setValue( "Apagar" , contagemTotal().get(2) );
        dataset.setValue( "Prevenir" , contagemTotal().get(3) );
        dataset.setValue( "Abastecimento A" , contagemTotal().get(4) );
        return dataset;
}
```

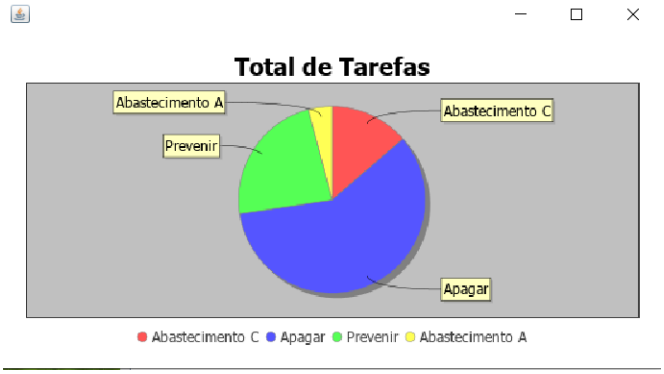

Figura 13: Total de Tarefas realizadas

7 Conclusão e perspetiva de trabalho futuro

Na primeira fase do trabalho prático foi realizada uma apresentação do estado da arte sobre os sistemas multiagente e sobre as técnicas usadas em EDM. Tendo em conta os factos apresentados, não restam dúvidas sobre a excelente aplicabilidade que os SMA têm em problemas deste domínio. Isto deve-se a conjunto de fatores nos quais os SMA são conhecidos por atuarem com distinção, tais como: a existência de diversas fontes de informação, a velocidade com que os dados devem ser processados para que obtenha uma resposta em tempo útil, a existência de diversas entidades que devem ser mapeadas para o domínio da solução, entre outros. O próprio conceito de agente enquanto entidade que perceciona um ambiente através de sensores, atuando com base nos dados recebidos de forma a modificar esse ambiente, vai de encontro ao que se pretende obter num sistema de suporte à decisão para EDM.

Tendo como finalidade a implementação de um SMA para EDM, foi também apresentada toda a documentação necessária para o caraterizar. No entanto, tentou-se que esta fosse o mais abstrata possível, dando espaço de manobra para tomar algumas decisões aquando do desenvolvimento do código. O algoritmo de alocação de recursos não foi ainda descrito, embora vários aspetos que este deve considerar tenham sido abordados ao longo do texto. Um algoritmo ótimo será aquele que minimize a área ardida, tendo como prioridade garantir que nenhuma célula do tipo habitação seja afetada. Este deve ainda procurar combater os fogos numa fase inicial, delegando para isso as unidades que conseguem deslocar-se até ao fogo em menor tempo. Na atribuição de tarefas aos agentes participativos devem ser considerados os recursos que estes apresentam, não delegando tarefas que os agentes não consigam cumprir. Caso um agente não apresente recursos suficientes mas exista um posto de abastecimento perto, o algoritmo deve também considerar estas situações. Ficou também por definir a forma de propagação de um incêndio e o momento em que este deve parar de crescer.

Na segunda fase do trabalho prático procedemos à implementação do sistema modelado na fase anterior. Foram analisadas diversas opções para a implementação do algoritmo de alocação de recursos, sendo que alocar por critério temporal pareceu ser uma opção viável, principalmente quando reforçada por autonomia por parte dos agentes participativos e a existência de um agente preventivo. No entanto, o sucesso do combate aos incêndios encontra-se condicionado pela localização dos postos de abastecimento. Em casos em que seja necessário os agentes percorrerem largas distâncias para reabastecer, a melhor opção não será enviar o agente mais rápido, mas sim o agente com mais capacidade de armazenar recursos.

De forma a aumentar a colaboração existente entre os agentes participativos, gostaríamos

de ter implementado um mecanismo que permitisse um agente apagar uma célula por onde este passasse e esta se encontrasse à arder, devendo posteriormente avisar o agente participativo, originalmente alocado a apagar aquela célula, que realizou a tarefa em sua vez.

A fluidez da interface também pode ser melhorada se for utilizada uma biblioteca orientada à construção de mapas de grelhas. No nosso caso, utilizamos Swing por ser uma ferramenta com a qual já tínhamos trabalhado. No entanto, surgiram algumas limitações tais como representar vários agentes participativos do mesmo tipo numa mesma célula do mapa e fazer deslocar os agentes de forma continua entre células.

Referências

[1] Zhou, L., Wu, X., Xu, Z., & Fujita, H. (2018). Emergency decision making for natural disasters: An overview. International journal of disaster risk reduction, 27, 567-576.

[2] Wooldridge, M. (1999). Intelligent Agents. In Weiss, G., editor, Multiagent Systems - A modern Approach to Distributed Artificial Intelligence. MIT Press.

[3] Novais, Paulo. Analide, Cesar.(Setembro 2006) Agentes Inteligentes - Texto Pedagógico. Grupo de Inteligência Aritificial. Centro de Ciências e Tecnologias de Computação.

[4] Balachandran, Bala M. "Developing Intelligent Agent Applications with JADE and JESS." Lecture Notes in Computer Science, 2019, pp. 236–244.

[5] Bergenti, Federico Vargiu, Eloisa. (2010). Multi-Agent Systems in the Industry. Three Notable Cases in Italy.. CEUR Workshop Proceedings. 621.

[6] McArthur, Stephen D. J., et al. "Multi-Agent Systems for Power Engineering Applications—Part I: Concepts, Approaches, and Technical Challenges." IEEE Transactions on Power Systems, vol. 22, no. 4, Nov. 2007, pp. 1743–1752.

[7] McArthur, Stephen D. J., et al. "Multi-Agent Systems for Power Engineering Applications—Part II: Technologies, Standards, and Tools for Building Multi-Agent Systems." IEEE Transactions on Power Systems, vol. 22, no. 4, Nov. 2007, pp. 1753–1759.

[8] M. Molina, G. Blasco, A multi-agent system for emergency decision support, Intell. Data Eng. Autom. Learn. (2003) 43–51.

[9] D. Moser, D. Pinto, A. Cipriano, Developing a multi-agent based decision support system for real time multi-risk disaster management, World Acad. Sci. Eng. Technol. 9 (3) (2015) 831–835.