4
Framework JASOF

Neste capítulo, descrevemos o framework JASOF (*Jadex Self-Organization Framework*) que tem como propósito fornecer uma base para construção de sistemas multi-agentes auto-organizáveis, proporcionando o reúso de soluções padronizadas. No primeiro momento, será apresentada a ideia geral do framework, em seguida, a sua arquitetura será explanada e por último serão mostrados os diagramas de caso de uso com suas respectivas descrições além dos diagramas de seqüência.

4.1
Visão Geral

Como já visto, uma importante barreira para a construção de sistemas auto-organizáveis é a falta de mecanismos previamente implementados e prontos para o uso. Atrelado a esse fator encontra-se também a pouca reusabilidade das soluções encontradas, que em boa parte dos casos são dependentes dos problemas a serem resolvidos. Frameworks lidam com este problema através do fatoramento de partes do software que já foram testadas e usadas em várias implementações de maneira que reduzem o custo de se produzir novos softwares aumentando a qualidade. Assim, um framework pode ser definido como sendo uma aplicação semi-completa e reutilizável que pode ser especializada para produzir aplicações específicas.

O framework JASOF visa servir como um mecanismo para construção de tais sistemas, oferecendo um modelo arquitetural e padrões de auto-organização prontos para o uso. Além de permitir a construção de sistemas auto-organizáveis, é possível também, através do framework, implementar novos padrões ainda não documentados na literatura. O fatoramento mencionado no parágrafo anterior fica por conta do uso dos padrões recorrentes nos mecanismos de coordenação.

Neste contexto, o JASOF provê uma representação de ambiente baseado na abordagem A&A, descrita na Seção 2.3, sendo composto por agentes e artefatos [Omicini et al., 2005]. O ambiente é definido como um conjunto de posições interligadas que formam um grid e cada posição contém uma localização que é gerenciada por um agente, como será visto em mais detalhes. Sendo assim, informações são armazenadas e retiradas desta localização através da troca de mensagens com este agente responsável e este mesmo agente também pode realizar determinados padrões disponíveis no framework.

Os padrões são disponibilizados através de um conjunto de planos e metas prontos para o uso, seguindo a abordagem Belief-Desire-Intetion (BDI), necessitando apenas a importação de determinados arquivos na construção dos agentes do sistema. Ao importar estes planos e metas, o agente torna-se hábil a realizar mecanismos de coordenação, ou seja, capacidade de reagir a determinados eventos no ambiente e também mecanismos de propagação, que é a capacidade de propagar um evento ao ambiente. Os padrões presentes são: Evaporation, Replication, Diffusion e Aggregation. Todos serão apresentados em detalhes na seção seguinte.

4.2 Arquitetura

Nesta seção, descreve-se a arquitetura do framework JASOF. Inicialmente o ambiente é explicado em detalhes, seguido pela descrição de como os padrões são implementados e disponibilizados.

Como ilustra a Figura 12, o framework tem disponíveis quatro padrões básicos, descritos em detalhes em [Gardelli et al, 2007], e já mencionados na Seção 2.4, onde o quinto padrão, *Collective Sort*, que foge um pouco do escopo do framework por ser um padrão de distribuição ordenada, foi alocado para futuras versões do framework. O padrão *Evaporation* é o mecanismo responsável pela eliminação da sobrecarga no ambiente, evitando informações em demasia e muitas vezes desnecessárias. Já o padrão *Aggregation* tem por finalidade diminuir a

informação redundante em áreas semelhantes, porém incrementando o seu fator de relevância ao meio. Por fim, com o padrão *Diffusion* é possível difundir uma determinada informação pelo ambiente, aplicando o decremento da relevância de acordo com a distância da fonte, diferentemente do padrão *Replication*, que tem o mesmo propósito, porém este faz uma cópia da informação às demais localizações receptoras.

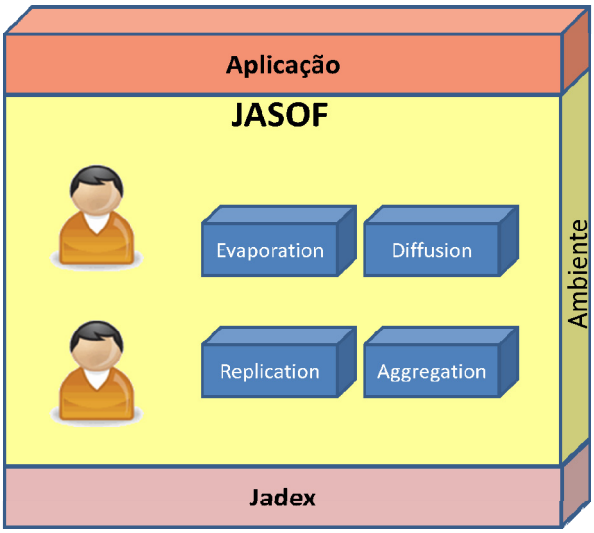

Figura 12 - Arquitetura do JASOF

O framework JASOF é baseado no paradigma BDI e foi implementado como uma extensão do framework Jadex, considerada uma das plataformas para criação de agentes de software – ver Seção 2.5 para mais detalhes. A escolha desta plataforma se deu devido ao fato da mesma possibilitar a criação de planos e metas encapsulados como *capabilities*. Este encapsulamento facilita a modularização e o acoplamento do sistema de agentes de software, pois a adição destes planos e metas ocorre em razão da importação da *capability* na estrutura do agente. Adicionalmente, o Jadex provê todos os mecanismos necessários para o desenvolvimento de acordo com o paradigma BDI.

No Jadex os agentes têm sua estrutura declarada em arquivos XML chamados Agent Definition File (ADF). Um ADF descreve aspectos como a base de conhecimento (*belief base*) do agente, quais mensagens o agente poderá receber ou enviar, seus objetivos e planos, dentre outros. Assim como agentes,

capabilities são declaradas em ADFs, descrevendo a ocorrência dos planos e o alcance das metas. Logo, para que um agente possa realizar os padrões disponibilizados pelo JASOF, basta importar a *capability* referente ao plano almejado no arquivo de descrição do agente.

Para facilitar o entendimento dos padrões disponibilizados no JASOF e sua utilização, estes são explicados em mais detalhes nas seções seguintes. Antes, porém, é feita uma explicação do funcionamento do ambiente no JASOF.

4.2.1 Ambiente

Nos sistemas auto-organizáveis naturais e nos protótipos dos artificiais o ambiente exerce um papel fundamental na dinamicidade do sistema, pois é através da interação com ele que a auto-organização de diversos sistemas é alcançada. Um exemplo típico é o caso das formigas, que realizam interações com o ambiente, como o armazenamento de feromônio e a sua percepção, e então, a partir dessas ações, guiam suas decisões e realizam seus trabalhos. Portanto, a modelagem do ambiente para o framework é peça fundamental na construção de tais sistemas, pois é a partir dela que os padrões de auto-organização são possíveis. Sendo assim, o ambiente disponível no framework JASOF é modelado como um conjunto de posições discretas e interligadas que formam um grid de duas dimensões, além disso, como já mencionado, é baseado na abordagem A&A, proporcionando-lhe uma característica de ambiente ativo.

Em cada posição do ambiente é alocada uma localização. Importante definir a diferença entre ambas: posição refere-se a um ponto de referência espacial no ambiente, enquanto localização é uma área localizada numa posição que pode ser habitada por diversos agentes. Em cada localização existe um agente responsável pelo seu gerenciamento, este agente é chamado de agente Location. É através do agente Location que o ambiente se torna ativo, ou seja, executa ações e interage com os demais participantes do sistema. Esses participantes são compostos tanto pelos agentes Locations das distintas localizações como pelos demais agentes do ambiente, esses últimos entendidos como agentes usuários.

Através da troca de mensagens com o agente Location de uma localização específica é possível a um agente realizar a inserção de informações na devida localização, bem como a leitura das informações ali contidas. Porém, observa-se a exigência do agente requisitante estar na mesma localização do agente Location. Toda comunicação com o agente Location para fins de inserção e leitura de dados é um ponto flexível no framework (*hot-spot*), porém a infraestrutura do ambiente, composta de localizações, posições e agentes Location, é um ponto fixo (*frozen-spot*). Ou seja, no quesito de comunicação, fica a cargo da implementação definir quais padrões o agente Location da respectiva localidade irá realizar, sendo assim, ao importar tais padrões o agente se torna hábil a entender determinadas mensagens e reagir a elas. Como exemplo, na Figura 13, que exibe parte da *capability* do padrão Diffusion, pode-ser observar que ao adicioná-la à sua definição, o agente passará a receber mensagens do tipo *diffusionMsg*.

```
<events>
   <messageeventref name="diffusionMsg" exported="true">
      <abstract/>
   </messageeventref>
</events>
```

Figura 13 - Evento do Padrão Diffusion

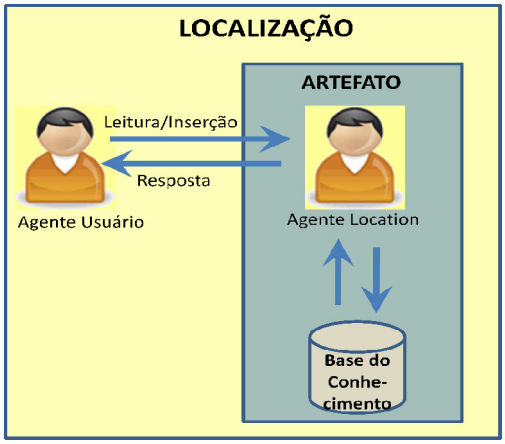

Figura 14 - Inserção/Leitura no Ambiente

A Figura 14 ilustra o mecanismo de inserção e leitura nas localizações. A razão pela abordagem de agentes Location em cada localização do ambiente se deu em virtude da busca por formas de descentralização, apesar do contra-argumento do elevado número de agentes e pela possível sobrecarga de comunicação para leitura e escrita de informações. Nas seções seguintes poder-se-

á observar a facilidade do ambiente e dos agentes realizarem os padrões de auto-organização.

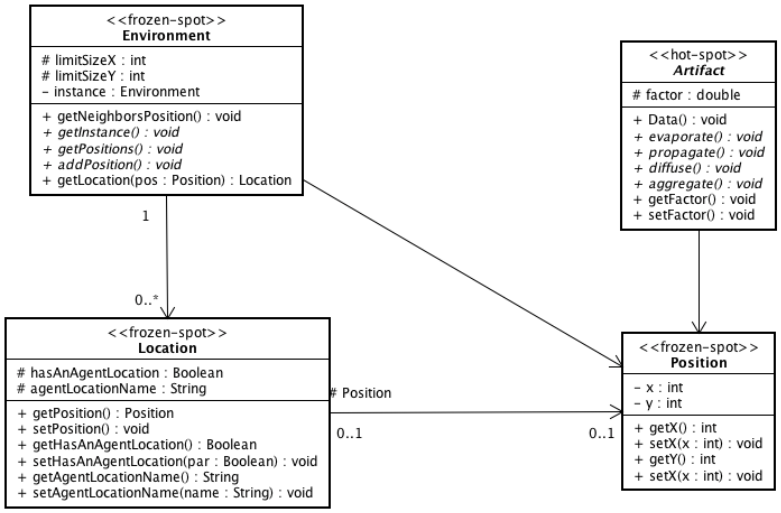

Figura 15 - Diagramas de Classes Ambiente

As informações armazenadas nas localizações são um ponto flexível do JASOF, através da classe *Data* (Figura 15) é possível realizar a extensão da mesma e desenvolver quais informações estão sendo manipuladas no ambiente, ou seja, quais dados serão lidos e escritos nas localizações. Ainda, é possível também definir restrições de leitura e escrita aos dados das localizações, mediante políticas desenvolvidas pela aplicação.

Seguindo o mesmo exemplo dado das formigas, o objeto feromônio representado no framework seria uma extensão da classe *Data*. Essa mesma classe também determina como será o impacto da realização dos padrões nas informações ali representadas, ou seja, na ocorrência do padrão Evaporation, por exemplo, como a informação irá evaporar? Este impacto em específico é determinado através do método *evaporate()* da classe *Data*. Entretanto, o mesmo ocorre para os demais padrões, através de seus respectivos métodos da classe. Sendo assim, para cada padrão um método é definido para realização do impacto:

- Padrão Evaporation, método *evaporate()*;
- Padrão Diffusion, método *diffuse()*;
- Padrão Aggregation, método *aggregate()*;

- Padrão Replication, método *replicate()*.

As informações supracitadas, representadas pela classe *Data*, e os agentes Location compõem o que na abordagem A&A é definido como artefato. Artefato são entidades passivas e reativas que proveem serviços ou funcionalidades a serem exploradas por outros agentes através de sua interface de uso. Sendo exatamente desta forma como ocorre no framework, interface caracterizada pelas mensagens de comunicação com o agente Location e os serviços pela possibilidade de inserir e ler as informações disponíveis na localização.

Entendido o funcionamento e a estrutura do ambiente no framework, nas seções seguintes serão descritos em mais detalhes o funcionamento dos padrões e como é possível implementá-los através do framework.

4.2.2 Padrões

A Figura 16 exibe o diagrama de classes dos planos dos padrões existentes no framework JASOF. A classe *PatternPlan* abstrai os planos dos padrões, sendo uma extensão da classe *Plan* oferecida pelo Jadex. É através desta classe que ocorre a integração dos planos elaborados no JASOF com os agentes construídos no Jadex. Cada padrão especializa a classe *PatternPlan*, e para o caso específico do Diffusion e Replication, estes são especializações da classe *PropagatePatternPlan*, seguindo o conceito do padrão de projeto *Strategy* [Gamma et al., 1995]. Ademais, cada padrão será descrito em detalhes nas seções seguintes, exibindo o seu propósito e sua forma de utilização.

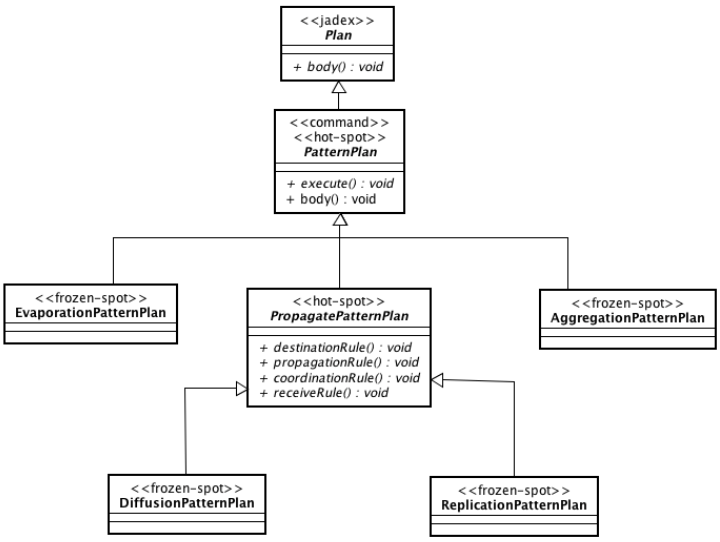

Figura 16 - Diagrama de Classes dos Padrões

4.2.2.1 Padrão Diffusion

Como já visto em capítulos anteriores, através do padrão Diffusion é possível difundir uma determinada informação pelo ambiente decrementada de uma porcentagem em sua relevância, sendo este valor proporcional à sua distância da fonte. Em sistemas naturais, quando o feromônio é depositado no ambiente, espontaneamente o mesmo tende a se difundir pelas localidades vizinhas, este processo é conhecido como difusão [Bonabeau et al., 1999]. Como exemplo, tem-se o perfume exalado por esses feromônios, que possui um determinado alcance e é difundindo igualmente nas localidades ao redor.

No framework JASOF, para que um agente possa realizar o padrão *Diffusion* este deve importar a *capability DiffusionPattern.capability.xml* ao seu arquivo de descrição (ADF). Na Figura 17 abaixo, que exibe o principal trecho desta *capability*, é possível observar qual mensagem fará o agente ser acionado e qual plano será utilizado. A mensagem encarregada de realizar a propagação da informação e o seu respectivo recebimento recebe o nome de *diffusionMsg*. Já o

plano em questão é definido através da classe *DiffusionPatternPlan*, que como exibe a Figura 16 acima, estende a classe *PropagatePatternPlan*.

```
<beliefs>
    <beliefsetref name="artefact" class="Artefact" exported="true">
        <abstract/>
    </beliefsetref>
    <!-- Locations Neighbors -->
    <beliefsetref name="neighbors" class="Position" exported="true">
        <abstract/>
    </beliefsetref>
</beliefs>
 <plans>
    <plan name="diffusion_plan">
        <body class="DiffusionPatternPlan"/>
        <waitqueue>
            <messageevent ref="diffusionMsg"/>
        </waitqueue>
    </plan>
 </plans>
 <events>
    <messageeventref name="diffusionMsg" exported="true">
        <abstract/>
    </messageeventref>
 </events>
```

Figura 17 - Trecho do DiffusionPattern.capability.xml

Esta última classe, *PropagationPatternPlan*, estabelece quatro mecanismos de extensibilidade, que são: (i) *receiveRule*; (ii) *coordinationRule*; (iii) *destinationRule*; e (iv) *propagationRule*. Através do mecanismo *receiveRule* é possível tratar os eventos geradores do plano em questão, como por exemplo, qual agente foi responsável pela mensagem enviada. Já o mecanismo *coordinationRule* determina quais ações serão tomadas em decorrência do evento gerador, ou seja, é a reação do agente em face a um evento. Por outro lado, os dois mecanismos restantes são utilizados para a geração de um evento. Com o *destinationRule* é possível determinar quais agentes irão receber a mensagem, por exemplo, apenas os vizinhos do tipo Location, e através do *propagationRule* a mensagem a ser propagada é construída e enviada.

Sendo assim, o *DiffusionPatternPlan* é uma realização da classe *PropagationPatternPlan*, implementando os mecanismos supracitados. No *receiveRule* é estabelecido o tratamento da mensagem *diffusionMsg*, exibida na Figura 17, presente na *capability* do padrão Diffusion. Informações como remetente e dados contidos na mensagem são extraídos e armazenados em variáveis. No *coordinationRule* é determinado que, ao receber uma mensagem de

difusão, o agente deve ou não salvar o seu conteúdo na sua base do conhecimento (*Belief Base*). Alterações podem ser feitas no intuito de reduzir ou ampliar a reação do agente, como por exemplo, ao receber uma mensagem de difusão propagá-la por mais um determinado número de vizinhos. O *destinationRule* tem em sua implementação padrão a seleção de todos os atuais vizinhos do agente em questão, logo, caso o agente realize uma difusão todos os seus vizinhos irão receber a mensagem. Por último, o *propagationRule* cria a mensagem *diffusionMsg* tendo como receptores os agentes definidos no *destinationRule*, insere os dados presentes na sua base do conhecimento e envia a mensagem.

Assim como todo plano do Jadex, a classe *DiffusionPatternPlan* deve implementar o método *body()* proveniente da classe *Plan*. Assim como mostra a Figura 18, é aplicado o padrão de projeto *Template Method* [Gamma et al., 1995], definindo a ordem de execução dos procedimentos associados ao padrão Diffusion.

```
@Override
public void execute(Data data) {
    IEvent mEvent = ReceiveRule(); // What starts the plan
    CoordinationRule(mEvent); // What a need to do for it?
    DestinationRule(); // Who will receive my message
    PropagationRule(); // How i will send my message

}
```

Figura 18 - Template Method da classe DiffusionPatternPlan

4.2.2.2 Padrão Evaporation

O intuito do padrão Evaporation é minimizar a possibilidade de sobrecarga de informações no ambiente, baseando-se num critério de tempo. A evaporação é um processo que pode ser observado diariamente, embora com diferentes implicações. Por exemplo, a partir da intensidade de um perfume sentido é possível deduzir a quantidade e a distância da fonte. No caso das formigas, a alta concentração de feromônio pode indicar um caminho a uma fonte de comida recente; por outro lado, sua presença reduzida pode significar que a fonte está esgotada ou que nada foi encontrado.

O padrão Evaporation tem uma característica peculiar, assim como o padrão da próxima seção, o Aggregation. Ambos são padrões locais, ou seja, realizam ações sob informações presentes em determinada localidade que atuam. O Evaporation é dirigido por ações do ambiente, já o Aggregation por ações dos agentes usuários. Por serem padrões de ações locais, ambos são destinados ao agente Location, visto que este é o agente que gerencia as informações depositadas no ambiente. Contudo, o framework não impede o uso em agentes usuários, apesar de fugir das abordagens que o utilizam.

Para que um agente possa realizar o padrão Evaporation, este deve importar a *capability EvaporationPattern.capability.xml*, que tem seu principal trecho exibido na Figura 19 abaixo. Como mostra a figura, os agentes terão um repositório de objetos do tipo *Data* na sua base do conhecimento, serão estes objetos que sofrerão a ação de evaporação. O mecanismo de funcionamento do padrão Evaporation consiste em um objetivo (*goal*) recorrente que é acionado a cada 6000 milissegundos, a cada vez que é disparado, o plano *EvaporationPatternPlan* é executado.

```
<beliefs>
    <beliefsetref name="data" class="Data" exported="true">
        <abstract/>
    </beliefsetref>
</beliefs>

 <goals>
    <performgoal recur="true" recurdelay="6000" name="evaporation" exported="true">
        <unique/>
    </performgoal>
 </goals>

 <plans>
    <plan name="evaporation_plan">
        <body class="EvaporationPatternPlan"/>
        <trigger>
            <goal ref="evaporation"></goal>
        </trigger>
    </plan>
 </plans>
```

Figura 19 - Trecho do EvaporationPattern.capability.xml

O plano *EvaporationPatternPlan* estende diretamente a classe *PatternPlan*, do framework, como pode ser visto na Figura 16. Quando executado, este plano percorre a base do conhecimento do agente, e para cada objeto *Data* armazenado, o método *evaporate()* é invocado. Ao ser acionado, este método decrementa um

valor pré-definido ao total do valor de relevância da informação. Sendo assim, a cada ciclo que o plano é executado, o fator de relevância de todas as informações armazenadas na base do conhecimento do agente executor será decrementada. Por fim, quando o fator de relevância alcança o valor zero a informação é eliminada. Portanto, é desta forma que o ambiente desempenha a evaporação das informações nele armazenadas, onde cada agente da localidade executa o plano sob as informações que armazena.

4.2.2.3 Padrão Aggregation

O padrão Aggregation é um mecanismo de reforço, também observado em atividades humanas. Por exemplo, em um sistema de recomendações, quanto mais qualificações positivas se recebe sobre um determinado serviço ou produto, maior será a crença e relevância do serviço ou produto quando necessitado. O padrão Aggregation atua espontaneamente, ou seja, ao ser adicionada uma informação no ambiente esta sofre uma agregação de imediato, onde informações semelhantes são unidas e percebidas como uma única informação, porém, com um fator de relevância maior. Quando usado com o padrão Evaporation, o padrão Aggregation permite ao desenvolvedor criar um mecanismo de *feedback loop* positivo/negativo, permitindo o desenvolvimento de sistemas baseados no conceito de computação autonômica [Kephart et al., 2003].

No framework, o padrão Aggregation tem implementação similar ao padrão Evaporation. Como já mencionado, este padrão é de uso apropriado ao agente Location, pois é este quem gerencia as informações da localidade e por ventura executa uma agregação de informações. Através da importação da *capability AggregationPattern.capability.xml*, o agente passa a desempenhar o padrão. O funcionamento é semelhante ao descrito no padrão Evaporation, em que, a cada 6000 milissegundos o plano, neste caso o *AggregationPatternPlan*, é acionado. O valor de 6000 milissegundos não é fixo, ele pode ser alterado conforme a necessidade da implementação, foi estabelecido apenas como um valor inicial. Ainda seguindo a ideia do padrão anterior, ao ser executado, o plano irá percorrer a base de conhecimento do agente e para cada informação (*Data*) armazenada, irá

compará-la às demais e existindo similaridade executara o método *aggregate()*, onde a partir de então unirá as informações transformando-as em apenas uma com um fator de relevância superior.

4.2.2.4
Padrão Replication

O padrão Replication é normalmente encontrado em sistemas naturais como forma de aumentar a sua segurança e robustez. Por exemplo, todas as células do corpo humano mantêm uma cópia local do DNA, permitindo a elas a possibilidade de se recuperar de pequenas mutações. Este mecanismo também é comumente aplicado em discos rígidos de servidores como forma de evitar perdas, técnica conhecida como Redundant Array of Independent Drives (RAID).

Em uma comparação entre os padrões Replication e Diffusion, a diferença reside no fato de que na replicação os dados não sofrem qualquer tipo de alteração, pois são repassados aos vizinhos da forma que estão armazenados. Por outro lado, na difusão, ao serem repassados, estes sofrem uma modificação em sua relevância, em que esta é decrementada, como visto na Seção 4.2.2.1. No framework, esta diferença reside no método da classe *Data*, que determina o processo por qual a informação passará ao se realizar o padrão. Para um agente desempenhar o padrão Replication, este deve importar a *capability ReplicationPattern.capability.xml*, seguindo os mesmos passos já descritos no padrões anteriores.

O plano acionado por esta *capability* é o *ReplicationPatternPlan*, que segue a mesma estrutura do *DiffusionPatternPlan*, sendo composto por quatro mecanismos de extensibilidade, que são: *receiveRule*, *coordinationRule*, *destinationRule* e *propagationRule*. Os quatro mecanismos têm implementação similar aos do padrão Diffusion, alterando apenas a mensagem a ser enviada e recebida, que neste caso é a *replicationMsg*.

4.3
Pontos Flexíveis do Framework

Os pontos flexíveis do framework são:

- Informações trafegadas no ambiente, capacidade de criação da informação a ser trafegada. A partir da classe Artifact é possível realizar a sua extensão criando novos tipos de dados a serem trafegados e armazenados pelos agentes.
- Composição de padrões para surgimento de novos padrões. Mediante características do Jadex é possível a inclusão de vários planos em um agente, tornando possível a elaboração de novos padrões compostos.
- Criação de novos padrões básicos. Ponto flexível possível a partir da extensão da classe PatternPlan e a criação do seu respectivo plano.
- Ações de coordenação e formas de propagação de cada padrão. Este ponto flexível é abordado em mais detalhes no caso de uso.
- Possibilidade de adicionar aos agentes os padrões que lhe são adequados. Nenhum padrão básico obriga a existência de outros padrões.
- A forma de comunicação dos agentes com o agente Location. As mensagens trocadas para armazenamento e leitura podem ser estendidas para adequar a solução mediante o uso do framework.

4.4 Diagrama de Casos de Uso

A Figura 20 exibe o diagrama de casos de uso do framework JASOF. As descrições de cada caso de uso são apresentadas em seguida.

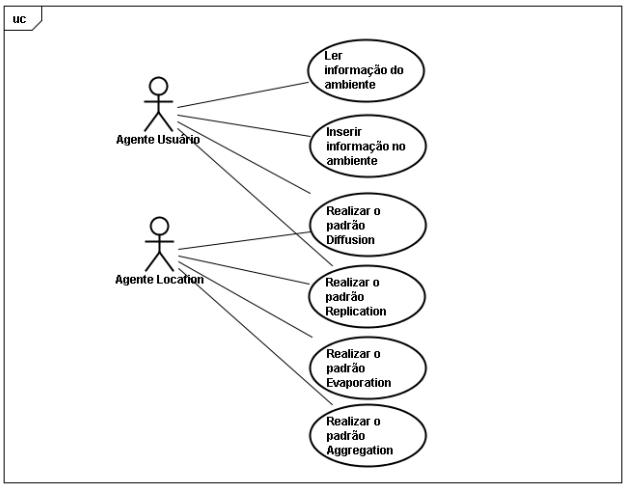

Figura 20 - Diagrama de Caso de Uso do JASOF

4.4.1 Realizar o padrão Diffusion

Descrição: Este caso de uso permite aos Agentes Usuário ou Location difundirem informações aos seus vizinhos no ambiente.

Atores: Agente Usuário ou Agente Location

Pré-Condições: Os agentes devem conter informações a serem enviadas em suas bases do conhecimento.

Pós-Condições: Os agentes da vizinhança terem recebido a informação, de relevância inferior, difundida pelo ator.

Fluxo Principal

1. Agente Usuário ou Location constrói a lista de destinatários da mensagem, a partir do conjunto de seus vizinhos, provido pelo ambiente;
2. Agente Usuário ou Location executa o processo pelo qual a informação sofre decremento em sua relevância;
3. Agente Usuário ou Location cria uma mensagem *diffusionMsg*, insere a informação manipulada e especifica os destinatários;
4. Agente Usuário ou Location envia a mensagem;

5. Os vizinhos destinatários recebem a mensagem *diffusionMsg*;
6. Os vizinhos realizam a ação de coordenação pelo recebimento da mensagem e retiram a informação contida na mesma e a salvam na sua base do conhecimento;
7. Caso de uso encerrado.

Fluxos Alternativos

(A1) Alternativo ao Passo 1: Nenhum vizinho é encontrado

1. O envio da mensagem é descartado;
2. Caso de uso encerrado.

4.4.2 Realizar o padrão Replication

Descrição: Este caso de uso permite aos Agentes Usuário ou Location replicarem informações aos seus vizinhos no ambiente.

Atores: Agente Usuário ou Agente Location

Pré-Condições: Os agentes devem conter informações a serem enviadas em suas bases do conhecimento.

Pós-Condições: Os agentes da vizinhança terem recebido uma cópia da informação armazenada pelo ator.

Fluxo Principal

1. Agente Usuário ou Location constrói a lista de destinatários da mensagem, a partir do conjunto de seus vizinhos, provido pelo ambiente;
2. Agente Usuário ou Location cria uma mensagem *replicationMsg*, insere a informação a ser enviada e especifica os destinatários;
3. Agente Usuário ou Location envia a mensagem;
4. Os vizinhos destinatários recebem a mensagem *replicationMsg*;

5. Os vizinhos realizam a ação de coordenação pelo recebimento da mensagem e retiram a informação contida na mesma e a salvam na sua base do conhecimento;
6. Caso de uso encerrado.

Fluxos Alternativos

(A1) Alternativo ao Passo 1: Nenhum vizinho é encontrado

1. O envio da mensagem é descartado;
2. Caso de uso encerrado.

4.4.3 Realizar o padrão Evaporation

Descrição: Este caso de uso permite aos Agentes Location realizarem o processo de evaporação das informações contidas na sua localização no ambiente. Por evaporar entende-se decrementar o fator de relevância da informação.

Atores: Agente Location

Pós-Condições: Informações contidas na localização do agente estarão com o fator de relevância decrementado. Se o fator de relevância for zero a informação deverá ser removida.

Fluxo Principal

1. De acordo com o especificado no padrão Evporation, um tempo para ocorrência de cada iteração é esperado, em seguida os demais fluxos são executados;
2. Agente Location recupera todas as informações contidas na sua base do conhecimento;
3. Agente Location executa o processo de evaporação para cada informação;
4. Caso de uso encerrado.

Fluxos Alternativos

(A1) Fluxo alternativo ao passo 3: Informação chegou a zero no fator de relevância

1. Agente Location remove a informação da sua localização caso o fator de relevância tenha chegado a zero;
2. Caso de uso continua no passo 3.

4.4.4
Realizar o padrão Aggregation

Descrição: Este caso de uso permite aos Agentes Location agregarem as informações semelhantes contida na sua localização no ambiente. Por agregar entende-se unir as duas informações em apenas uma com fator de relevância superior.

Atores: Agente Location

Pós-Condições: Informações semelhantes contidas na localização serão unidas, resultando em uma única informação com fator de relevância superior.

Fluxo Principal

1. De acordo com o especificado no padrão Aggregation, um tempo para ocorrência de cada iteração é esperado, em seguida os demais fluxos são executados;
2. Agente Location recupera todas as informações contidas na sua base do conhecimento e verifica se existem informações semelhantes;
3. Para cada grupo de informações semelhantes, o Agente Location executa o processo de agregação, o resultado, que é uma informação única com relevância superior, é salva na base do conhecimento;
4. Caso de uso encerrado.

4.5 Diagramas de Seqüência

Nesta seção apresentamos os diagramas de sequência referente aos casos de uso descritos na seção anterior, que são: Realizar o padrão Diffusion; Realizar o

padrão Replication; Realizar o padrão Evaporation e Realizar o padrão Aggregation.

4.5.1
Realizar o padrão Diffusion

A Figura 21 exibe o diagrama de sequência que mostra o cenário no qual um agente Location executa o padrão Diffusion.

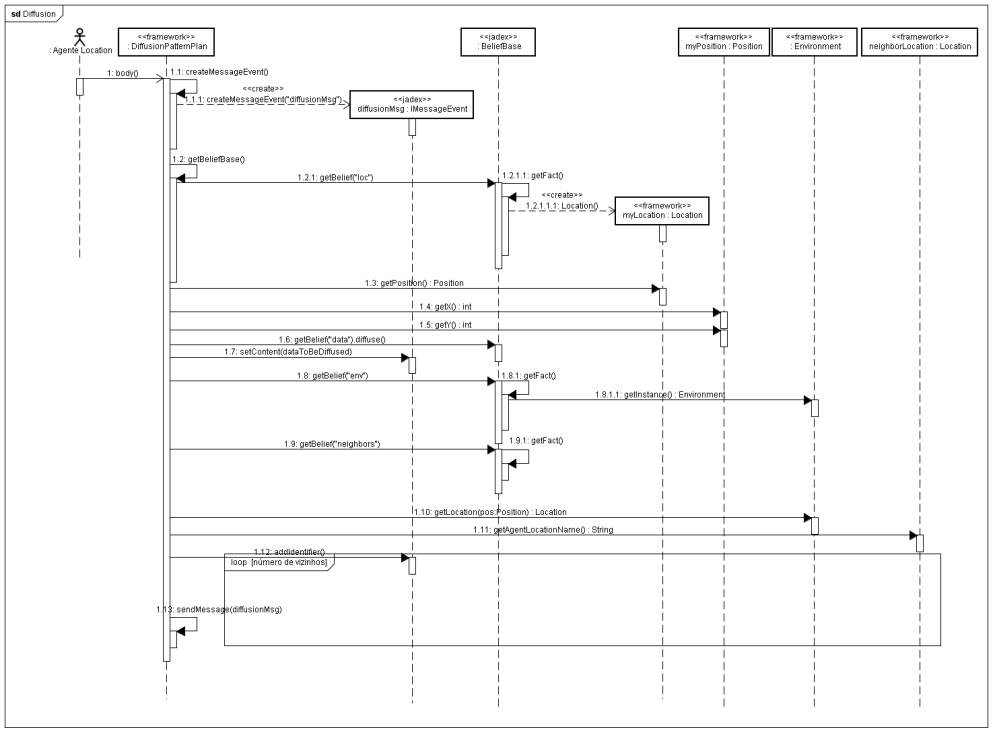

Figura 21 - Diagrama de Sequência do Padrão Diffusion

4.5.2
Realizar o padrão Replication

A Figura 22 mostra o diagrama de sequência da realização do padrão Replication pelo agente Location.

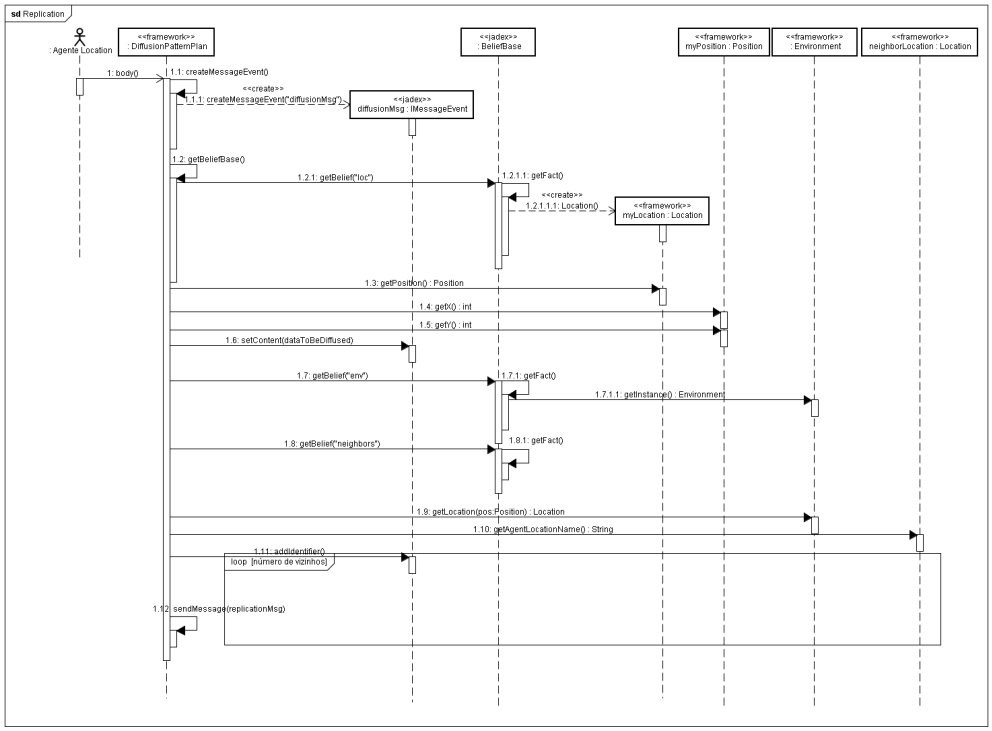

Figura 22 - Diagrama de Sequência do Padrão Replication

4.5.3
Realizar o padrão Evaporation

A Figura 23 apresenta o diagrama de sequência da realização do padrão Evaporation, onde os dados da localização sofrem a ação do processo *evaporate*.

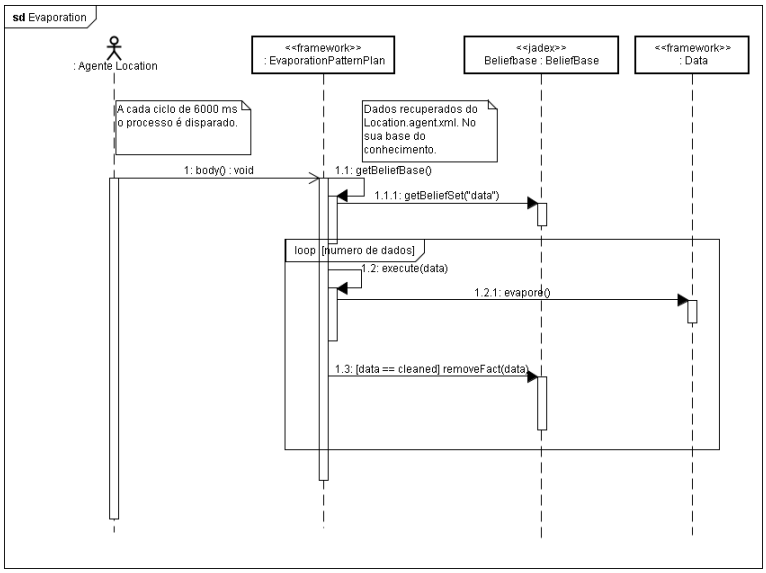

Figura 23 - Diagrama de Sequência do Padrão Evaporation

4.5.4
Realizar o padrão Aggregation

A Figura 24 exibe o diagrama de sequência do padrão Aggregation, onde as informações semelhantes da localização são agregadas, resultando uma informação de superior relevância.

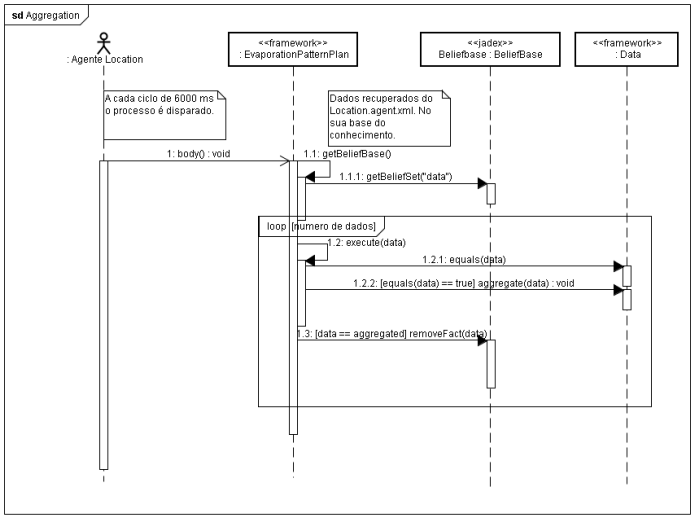

Figura 24 - Diagrama de Sequência do Padrão Aggregation