Pontifícia Universidade Católica
do Rio de Janeiro

Felipe Coimbra Bacelar

Uma abordagem baseada em gerenciamento de interesses para o particionamento dinâmico de simulações distribuídas

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Informática da PUC-Rio.

Orientador: Prof. Carlos José Pereira de Lucena

Rio de Janeiro
Agosto de 2013

Pontifícia Universidade Católica
do Rio de Janeiro

Felipe Coimbra Bacelar

Uma abordagem baseada em gerenciamento de interesses para o particionamento dinâmico de simulações distribuídas

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Informática do Centro Técnico e Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Carlos José Pereira de Lucena
Orientador
Departamento de Informática - PUC-Rio

Prof. Bruno Feijó
Departamento de Informática - PUC-Rio

Prof. Pierre Bommel
Departamento de Informática - PUC-Rio

Prof. José Eugênio Leal
Coordenador Setorial do Centro Técnico Científico - PUC-Rio

Rio de Janeiro, 08 de agosto de 2013

Felipe Coimbra Bacelar

Graduou-se no Curso de Bacharelado em Informática da Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio) em 2009.

Ficha Catalográfica

Bacelar, Felipe Coimbra

Uma abordagem baseada em gerenciamento de interesses para o particionamento dinâmico de simulações distribuídas / Felipe Coimbra Bacelar; orientador: Carlos José Pereira de Lucena. PUC-Rio, Rio de Janeiro, 2013.

58 f. : il. ; 29,7cm

Dissertação (Mestrado em Informática)–Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2013.

Inclui referências bibliográficas.

1. Informática – Tese. 2. Particionamento Dinâmico. 3. Simulação Distribuída. 4. Sistemas Multiagentes. 5. Gerenciamento de Interesses. I. Lucena, Carlos José Pereira de. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

A Deus, a meus pais, Antonio Carlos Coimbra Bacelar e Aniversina Gomes de Almeida, e a minha esposa Vanessa Carmo Marques Bacelar

Agradecimentos

Agradeço ao meu orientador, Carlos José Pereira de Lucena, pela oportunidade de desenvolver este trabalho. Foi uma honra ser aluno de um pesquisador que admiro tanto.

Agradeço ao professor Pierre Bommel por sua colaboração ao longo deste mestrado. Seu conhecimento teve papel muito importante nesta caminhada.

Agradeço aos demais professores do Departamento de Informática que contribuíram com minha formação acadêmica ao longo destes dois anos.

Agradeço a meus pais, Antonio Carlos Coimbra Bacelar e Aniversina Gomes de Almeida, pelo cuidado e dedicação em todos os momentos de minha vida.

Agradeço a minha esposa, Vanessa Carmo Marques Bacelar, por seu amor, carinho e compreensão.

Agradeço a meus irmãos, Rafael Carlos Coimbra Bacelar e Samuel Coimbra Bacelar, pelo suporte e incentivo durante este desafio.

Agradeço a meus familiares e amigos pelo apoio e por tornarem inestimáveis os poucos momentos de descontração nestes dois anos de mestrado.

Agradeço ao Tecgraf pelo apoio financeiro que permitiu a realização deste mestrado.

Agradeço aos meus companheiros do Tecgraf que tornaram esses anos menos difíceis, mantendo o ambiente de trabalho sempre agradável.

Agradeço principalmente a Deus por tornar tudo isto possível.

Resumo

Bacelar, Felipe Coimbra; Lucena, Carlos José Pereira de. **Uma abordagem baseada em gerenciamento de interesses para o particionamento dinâmico de simulações distribuídas**. Rio de Janeiro, 2013. 58p. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Para que simulações distribuídas baseadas em agentes possam ter alto grau de escalabilidade é necessário evitar gargalos de comunicação. Existe troca de mensagens entre máquinas toda vez que um agente contido em um determinado computador precisa interagir com elementos que se encontram em outro computador. O presente trabalho propõe particionar dinamicamente uma simulação de forma a manter um agente no mesmo nó da rede em que se encontram os elementos com os quais ele mais interage, reduzindo o custo de comunicação entre os computadores da rede. Para isto, é utilizado o conceito de gerenciamento de interesses, que visa prover ao agente apenas o conjunto mínimo de informações para que ele possa interagir com o ambiente de forma coerente. Para ilustrar a solução proposta foi desenvolvido um estudo de caso que compreende uma simulação distribuída representando um cenário de derramamento de petróleo no mar.

Palavras-chave

Particionamento Dinâmico; Simulação Distribuída; Sistemas Multiagentes; Gerenciamento de Interesses; Balanceamento de Carga

Abstract

Bacelar, Felipe Coimbra; Lucena, Carlos José Pereira de (Advisor). **An interest management approach to dynamic partitioning distributed simulations**. Rio de Janeiro, 2013. 58p. MSc. Dissertation - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

To achieve high scalability in distributed simulations is necessary to avoid communication bottlenecks. Messages between machines are necessary when an agent kept in a specific computer needs to interact with elements kept in another computer. This work presents an approach to dynamically partitioning a distributed simulation keeping each agent in the same network node where are the elements more accessed by it, reducing the communication cost between the network computers. To reach this objective, we are using the concept of interest management, which aims to provide to an agent only the smallest set of information necessary to allow it to interact with the environment in a coherent way. To illustrate the proposed solution was developed a case study comprehending a distributed simulation representing an oil spill scenario.

Keywords

Dynamic Partitioning; Distributed Simulation; Multi-agent Systems; Interest Management; Load Balancing

Sumário

Lista de Figuras

Lista de Tabelas

1
Introdução

Agentes podem ser entendidos como entidades autônomas localizadas em um ambiente. Eles são capazes de se comunicar entre si e interagir com seu ambiente. [Wooldridge e Jennings 1995]. Sistemas multiagentes podem ser compostos por conjuntos de agentes que interagem com o propósito de resolver um problema ou alcançar um objetivo. Em certos casos, os agentes colaboram entre si para atingir uma meta em comum. Em outros, os objetivos dos agentes podem ser opostos [Parunak et al. 2003].

O comportamento autônomo dos agentes e sua capacidade de tomada de decisão são fatores que tornam a utilização de sistemas multiagentes muito atrativa para o desenvolvimento de simulações.

Em nosso contexto é possível definir simulação como uma representação do comportamento de um sistema, real ou imaginário, através do tempo [Fujimoto 2000]. O emprego de simulações tem sido uma forma muito eficaz de estudar e analisar problemas reais. A possibilidade de modelar diferentes cenários para o mesmo problema e reproduzi-los diversas vezes permite a identificação de situações incomuns e comportamentos inesperados.

No entanto, como os agentes costumam ser unidades complexas de software, manter um número elevado deles em uma simulação pode ser muito custoso. Simulações de larga escala, com centenas ou milhares de agentes, tendem a ter seu desempenho comprometido.

Considerando o alto grau de paralelismo inerente ao uso de agentes, uma possível solução para o problema do alto custo computacional é distribuir a simulação em uma rede de processadores [Logan e Theodoropoulos 2001]. No entanto, a distribuição introduz problemas próprios como o *overhead* de comunicação entre os processadores.

Para reduzir o custo de comunicação entre os computadores envolvidos na distribuição, um algoritmo de Gerenciamento de Interesses deve ser empregado [Wang, Lees e Cai 2012]. O princípio deste modelo é baseado na idéia de que uma

entidade da simulação raramente usa todas as informações disponíveis, mas pode manifestar interesse em um subconjunto de informações que considere relevantes. Contudo, esse subconjunto pode mudar com o tempo. Ao se mover pelo ambiente, um agente pode desviar de um obstáculo e passar a ter um campo de visão maior ou pode se tornar visível para outros agentes, por exemplo. O mecanismo de Gerenciamento de Interesses deve ser capaz de se adaptar a estas mudanças.

De forma geral, em simulações baseadas em agentes, o conceito de interesse está fortemente associado à localização do agente no ambiente da simulação. Considera-se que agentes próximos tendem a se interessar um pelo outro e devem trocar informação de estado, visto que possuem maior probabilidade de interagir entre si.

O presente trabalho propõe utilizar uma técnica dinâmica de Gerenciamento de Interesses para particionar uma simulação de forma a manter agentes que interagem freqüentemente uns com os outros na mesma partição e garantir que estas partições se adaptem automaticamente às mudanças no estado da simulação. O objetivo é reduzir o custo de comunicação entre os computadores aumentando o desempenho geral da simulação.

1.1. Limitações das Abordagens Atuais

Há diversos trabalhos voltados para o estudo de gerenciamento de interesses em simulações e ambientes virtuais distribuídos na literatura [Morse 1996], [Bezerra, Cecin e Geyer 2008], [Lees et al. 2004], [Logan e Theodoropoulos 2001], [Theodoropoulos e Logan 2001], [Minson e Theodoropoulos 2005], [Morgan, Lu e Storey 2005] entre outros.

O assunto é mais amplamente aplicado na área de jogos massivos online, em que milhares de jogadores interagem simultaneamente no mesmo ambiente virtual e em outras áreas também relacionadas a ambientes virtuais distribuídos.

Embora a indústria de jogos eletrônicos explore bastante o tema, grande parte do conhecimento e das principais técnicas utilizadas não é publicada, pois pode ser traduzida em vantagem competitiva [Machado, Feijó e Kozovits 2006], [Strassburger, Schulze e Fujimoto 2008].

No que se refere à utilização deste tipo de técnica para particionar simulações baseadas em sistemas multiagentes especificamente, a quantidade de publicações expressivas é bastante reduzida.

Entre alguns dos principais trabalhos relacionados destacam-se: [Wang et al. 2009], [Wang, Lees e Cai 2012], [Logan e Theodoropoulos 2001], [Lees et al. 2004] e [Oguara et al. 2005].

Em [Wang et al. 2009] e [Wang, Lees e Cai 2012] considera-se que agentes próximos que possuem o mesmo comportamento tendem a interagir agora e no futuro. Propõe-se utilizar, além da informação de posicionamento de um agente, seu estado e sua velocidade para predizer agrupamentos de agentes baseados em suas metas. A abordagem apresentada nestes trabalhos é voltada para o particionamento de simulações de multidões e procura prever o agrupamento dos agentes usando fatores que não se aplicam de forma generalizada a qualquer tipo de simulação.

Em [Logan e Theodoropoulos 2001], [Lees et al. 2004] e [Oguara et al. 2005] é proposta a utilização do conceito de esferas de influência para decompor o estado de uma simulação e distribuí-la através de uma estrutura de dados própria que visa melhorar o balanceamento de carga. O estado da simulação é gerenciado através de variáveis que são distribuídas entre os computadores da rede. As variáveis de estados são mantidas em máquinas diferentes das que mantém os agentes. As variáveis mais acessadas por um agente devem ser mantidas nas máquinas mais próximas à máquina que mantém o agente.

Nesta solução, ao surgir a necessidade de reorganização das variáveis, em vez de reordenar parcialmente ou distribuir gradualmente as mesmas, todo o estado da simulação é reordenado, gerando um custo alto de processamento e comunicação que pode anular grande parte do ganho obtido com o processo.

1.2. Solução Proposta e Principais Contribuições

O objetivo do presente trabalho é melhorar o desempenho geral de uma simulação distribuída de sistemas multiagentes através da redução do custo de comunicação entre os processadores envolvidos. Para isto, é proposta uma abordagem para o particionamento da simulação de forma a manter cada agente no mesmo nó da rede em que se encontram os elementos com os quais ele mais

interage. Considerando o dinamismo da simulação, o conjunto de elementos mais acessados por um agente pode mudar ao longo do tempo. O particionamento deve refletir estas mudanças. Além disto, o método proposto se preocupa também em melhorar o balanceamento de carga, distribuindo eficientemente a carga de trabalho entre os computadores da rede.

A principal contribuição do trabalho proposto é uma nova abordagem para o particionamento dinâmico de simulações baseadas em agentes. Como contribuições secundárias destacam-se: o fato de a solução ser genérica, podendo ser utilizada em qualquer tipo de simulação, e a implementação de um estudo de caso para verificar as propriedades apresentadas.

1.3. Organização do Documento

O presente trabalho é dividido em sete capítulos contando o introdutório. Abaixo, segue uma breve descrição de cada um dos seis capítulos restantes desta dissertação:

No capítulo 2 são apresentados alguns conceitos e temas essenciais para o entendimento do trabalho proposto. Entre os assuntos comentados estão: sistemas multiagentes, simulações distribuídas e gerenciamento de interesses.

No capítulo 3 é descrito o trabalho proposto. São destacados a técnica de agrupamento dos agentes e o algoritmo de particionamento dinâmico da simulação.

O capítulo 4 descreve o estudo de caso implementado, uma simulação distribuída de derramamento de petróleo. Neste capítulo são apresentados os modelos da simulação e da solução final (modelo da simulação incluindo as classes responsáveis pelo método de distribuição) e também os *frameworks* utilizados para o desenvolvimento do projeto.

O capítulo 5 apresenta os resultados de um experimento realizado para verificar a viabilidade da solução proposta.

No capítulo 6 são analisados alguns trabalhos relacionados ao tema desta dissertação e são comentadas algumas técnicas que foram adotadas pelo presente trabalho.

O capítulo 7 expõe as conclusões do presente trabalho apontando suas contribuições, limitações e listando atividades que merecem atenção em trabalhos futuros.

2
Fundamentação Teórica

Este capítulo apresenta alguns conceitos essenciais para compreensão do presente trabalho. Entre alguns dos tópicos abordados estão: sistemas multiagentes, simulação distribuída e gerenciamento de interesses.

2.1. Sistemas Multiagentes

Seguem algumas características básicas de agentes segundo Wooldridge e Jennings (1995):

Autonomia: agentes podem operar sem interferência externa. Não há necessidade de interação humana ou de um controle central.

Reatividade: agentes são capazes de efetuar mudanças no ambiente e de reagir a estas mudanças.

Proatividade: agentes podem tomar iniciativa em vez de apenas reagir a eventos. São geralmente orientados a objetivos.

Interatividade: agentes possuem habilidade social. São capazes de se comunicar com outros agentes. Podem também perceber mudanças em seu ambiente e agir sobre ele.

Segundo Wooldridge (2002), sistemas multiagentes são sistemas compostos por conjuntos de agentes que interagem entre si.

Em sistemas multiagentes, geralmente os agentes interagem com o propósito de solucionar um problema ou atingir um objetivo. Em alguns casos, os agentes colaboram entre si para alcançar uma meta em comum. Em outros casos, os objetivos dos agentes são opostos [Parunak et al. , 2003].

Agentes percebem o ambiente e outros agentes através de sensores e efetuam alterações através de seus atuadores ou efetuadores. Abaixo segue um esquema exemplificando o funcionamento de um agente:

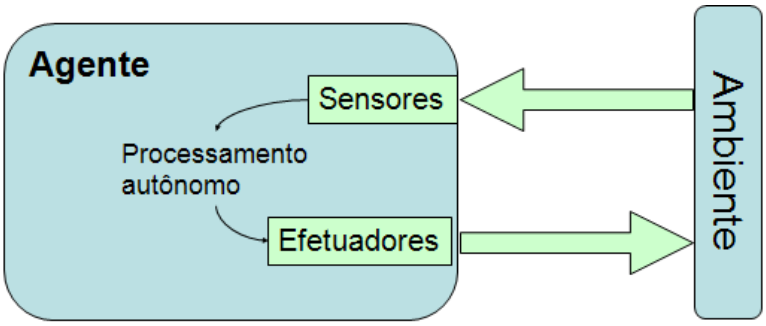

Figura 1 – Interação de um agente com o ambiente

Cada agente possui seus próprios sensores e atuadores. Agentes diferentes possuem sensores diferentes com alcances diferentes.

2.2. Simulação Distribuída

Segundo Fujimoto (2000) simulação é um modelo que representa o comportamento de um sistema real ou imaginário através do tempo. O autor aponta que simulações têm sido amplamente utilizadas para analisar comportamentos de sistemas sem necessariamente construí-los, visto que o custo e o risco de construir protótipos podem ser altos.

Uma grande vantagem do uso de simulações está na possibilidade de reproduzir experiências alterando parâmetros, condições e elementos para estudar os efeitos de cada alteração.

Outra importante forma de utilização de simulações destacada pelo autor é criação de mundos virtuais em que humanos ou dispositivos físicos podem interagir. Um exemplo é o uso de simulações de vôo para treinamento de pilotos.

Simulação distribuída refere-se à tecnologia que permite que simulações sejam executadas em sistemas compostos por múltiplos computadores, como uma rede doméstica ou até mesmo a Internet.

2.2.1. Benefícios

Fujimoto sugere quatro benefícios principais do uso de simulações distribuídas:

Redução do tempo de execução: dividir a simulação em processos que possam ser executados paralelamente e executar estes processos em N computadores diferentes, pode reduzir o tempo de execução em até N vezes.

Reduzir o tempo de execução é importante para que a simulação possa se tornar útil em tomadas de decisão estratégicas [Fujimoto 2000].

Distribuição geográfica: executar a simulação em computadores geograficamente distribuídos permite a criação de mundos virtuais em que pessoas localizadas em diversas partes do mundo podem se comunicar e interagir poupando o custo e o tempo de longas viagens [ibid.].

Aplicações práticas para visualizar o conceito são encontradas em simulações militares para treinamento, em jogos massivos online e na área de ensino à distância [Strassburger, Schulze e Fujimoto 2008].

Integração de simuladores que executam em máquinas de diferentes fabricantes: a integração de simuladores construídos por diferentes fabricantes para formar um único ambiente de simulação exige que a execução seja efetuada de forma distribuída [Fujimoto 2000]. Segundo Fujimoto, é menos custoso unir os simuladores existentes, cada um executando em uma máquina diferente, para criar um novo mundo virtual do que portar todos os simuladores para um único computador.

Este fator é importante pois [Strassburger, Schulze e Fujimoto 2008] aponta a tendência de cooperação entre empresas de diferentes segmentos para a criação de *clusters* que permitam a execução de simulações de larga escala.

Tolerância a falhas: a utilização de diversos computadores aumenta consideravelmente a possibilidade do sistema se recuperar de falhas. Caso um processador deixe de funcionar ou seja desconectado da rede, outro processador pode assumir a carga de trabalho da máquina que apresentou problemas. Em execuções que utilizam apenas um computador, se este parar de funcionar, toda a simulação pára [Fujimoto 2000].

2.2.2. Panorama Atual

De acordo com [Strassburger, Schulze e Fujimoto 2008], simulações distribuídas, apesar de serem amplamente utilizadas na esfera militar, ainda possuem pouca penetração na indústria.

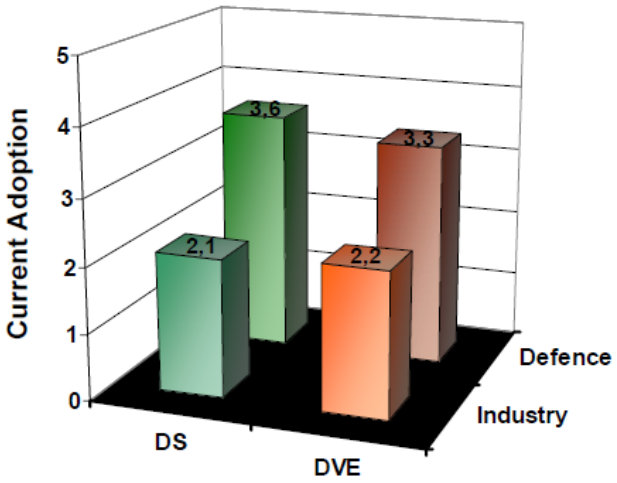

Figura 2 – Adoção de simulações distribuídas e ambientes virtuais distribuídos. Extraído de [Strassburger, Schulze e Fujimoto 2008]

A figura 2 deixa clara a diferença entre o setor de defesa e a indústria em relação à adoção de simulações distribuídas e ambientes virtuais distribuídos. As instituições militares têm utilizado este tipo de aplicação principalmente em áreas como treinamento e aquisição. Desta forma é possível realizar treinamento de táticas militares sem o custo e o risco de ir a campo. Além de ser possível verificar o funcionamento de certos tipos de produtos e componentes antes de efetivamente comprá-los.

O estudo destaca alguns domínios que podem ser explorados para ampliar a utilização de simulações distribuídas e ambientes virtuais distribuídos pela indústria:

Sistemas de suporte a decisão para controle de crises e catástrofes: entre as possíveis aplicações encontram-se simulações de efeitos de uma crise e simulações de técnicas e medidas de resposta a uma catástrofe, como, por exemplo, operações de resgate.

Aplicações de treinamento virtual: como se trata de um campo de atuação de sucesso no meio militar, acredita-se que seja natural sua adoção pela indústria, principalmente em treinamentos que envolvam ambientes geograficamente distantes, como filiais internacionais.

Reuniões virtuais: reuniões em ambientes virtuais distribuídos permitem interação social e podem ser utilizadas para pesquisa, negócios ou entretenimento.

Simulações de cadeias de suprimentos industriais: a principal motivação do uso de simulações distribuídas neste caso é a necessidade de proteção da

propriedade intelectual. Membros de uma cadeia de suprimentos podem adicionar seus módulos à simulação como uma caixa preta, sem revelar informações de seus processos internos.

2.2.3. Mudanças de Estado e Passagem de Tempo

Mudanças de estado da simulação ocorrem com o passar do tempo. Fujimoto (2000) aponta duas maneiras de classificar simulações com respeito ao mecanismo de fluxo de tempo. De acordo com o autor, simulações podem ser contínuas ou discretas. Em simulações contínuas, o estado da simulação muda continuamente através do tempo. O comportamento do sistema é modelado através de equações que descrevem como o sistema muda em função do tempo da simulação. Em simulações discretas, o estado muda apenas em pontos discretos no tempo da simulação.

Segundo Fujimoto (2000) as mudanças de estado em simulações distribuídas discretas podem ocorrer através de dois tipos de mecanismo de fluxo de tempo:

Time-Stepped: o tempo da simulação é dividido em passos de tempo de tamanho idêntico e o avanço da simulação ocorre de um passo para o próximo. A cada passo de tempo, o estado da simulação é atualizado.

Ações que ocorrem no mesmo passo de tempo são considerados simultâneas. Se uma ação depende da conclusão de outra para ocorrer, ambas não podem pertencer ao mesmo passo de tempo. Geralmente ações que ocorrem no mesmo passo de tempo não possuem efeito uma na outra dentro do passo de tempo em questão. Isto permite paralelizar as execuções destas ações de forma que cada uma seja executada em um computador.

Event-Driven: um evento é uma abstração usada para modelar alguma ação instantânea no ambiente da simulação. Um evento geralmente altera o estado de um ou mais elementos da simulação. Em vez de atualizar o estado da simulação a cada passo de tempo, é mais eficiente atualizá-lo cada vez que um evento acontece. Cada evento possui uma estampa de tempo associada. O tempo da simulação não avança de um passo de tempo para o próximo. Em vez disto, avança da estampa de tempo de um evento para a estampa de tempo do próximo evento.

2.3.Gerenciamento de Interesses

Informar a todas as entidades sobre todos os elementos da simulação é uma prática custosa e desnecessária. De forma geral, grande parte da informação é desperdiçada, gerando custo de comunicação desnecessário e problemas de escalabilidade.

Em uma simulação distribuída de larga escala, raramente uma entidade precisa receber informação sobre todos os elementos do ambiente. Um agente apenas necessita de informação sobre os elementos com os quais pode interagir ou exercer influência.

Com o intuito de prover aos agentes apenas informações relevantes, reduzindo assim o custo de comunicação, foi criado o conceito de Gerenciamento de Interesses [Morse 1996].

Segundo Morse, neste modelo os agentes expressam os dados que consideram relevantes através de *Interest Expressions* (IE, ou “expressões de interesse” em tradução livre). Uma IE é a especificação dos dados que uma entidade precisa receber de outra para interagir com ela corretamente. O modelo prevê também entidades chamadas *Interest Managers* (IM, ou “gerentes de interesse” em tradução livre), responsáveis por receber as IEs dos agentes e filtrar as informações em subconjuntos que atendam as necessidades dos agentes. Neste contexto, o dado de interesse de um agente é chamado *Domain of Interest* (DOI, ou “domínio de interesse” em tradução livre). O domínio de elementos que são de responsabilidade de um IM é chamado *Domain of Responsibility* (DOR, ou “domínio de responsabilidade”). Tanto DOI quanto DOR são domínios multidimensionais e não estão restritos apenas à área geográfica ou região, embora estes sejam os casos mais comum de observar. A autora destaca que IEs são freqüentemente auto-referenciais. Uma entidade pode estar interessada em todas as entidades dentro de um certo raio a partir de si própria, por exemplo. Ressalta também que interesses possuem alcances diferentes para entidades diferentes.

Métodos de gerenciamento de interesses costumam ser classificados na literatura em duas categorias:

Regiões: o ambiente da simulação é dividido igualmente em regiões e cada uma destas é atribuída a um servidor. Todos os agentes enquadrados em uma

região são atribuídos ao servidor responsável pela mesma. Se um agente muda de região no ambiente virtual, ele é migrado para o servidor responsável pela nova região [Morgan, Lu e Storey 2005],[Bezerra, Cecin e Geyer 2008], [Wang, Lees e Cai 2012].

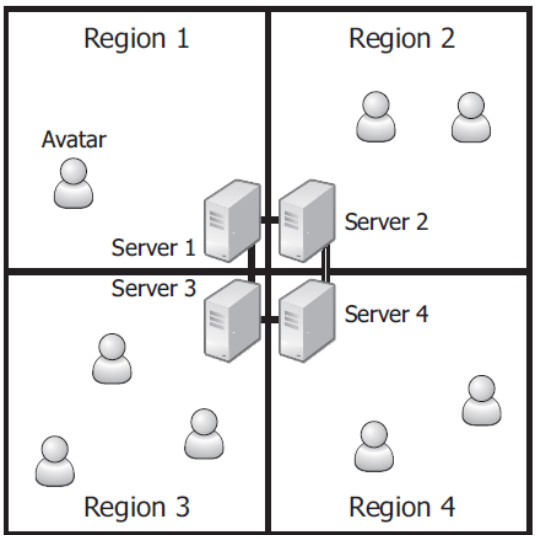

Figura 3 – Divisão por regiões. Extraído de [Bezerra, Cecin e Geyer 2008]

Esta abordagem pode ter bons resultados para simulações com poucos elementos móveis. Em casos de simulações em que as entidades tendem a se agrupar, esta estratégia pode gerar desequilíbrio de carga [Wang et al. 2009]. A divisão por região é uma estratégia de particionamento estático muito adotada em simulações distribuídas de sistemas multiagentes. No entanto, em muitos casos, esta técnica é adotada por sua simplicidade e não por alguma observação significativa [Wang et al. 2009], [Wang, Lees e Cai 2012].

Área de Interesse: basicamente o termo área de interesse é caracterizado por uma área ao redor da entidade que determina o conjunto de elementos com os quais a entidade pode interagir (ler ou atualizar). Uma entidade pode interagir apenas com os elementos que estão dentro de sua área de interesse, necessitando de informação apenas sobre estes elementos [Morgan, Lu e Storey 2005], [Bezerra, Cecin e Geyer 2008], [Wang, Lees e Cai 2012], [Wang et al. 2009].

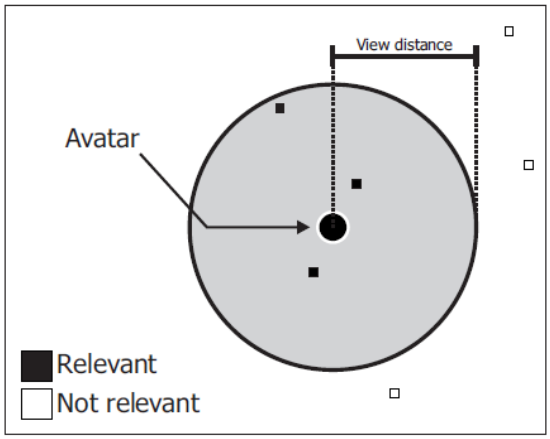

Figura 4 – Área de Interesse. Extraído de [Bezerra, Cecin e Geyer 2008]

Algumas variações do conceito podem ser encontradas na literatura. Entre elas estão:

Área de interesse baseada em campo de visão: apenas os objetos dentro do campo de visão de uma entidade compõem sua área de interesse. Elementos que estejam atrás da entidade ou de um obstáculo são descartados, mesmo que estejam dentro do raio que determina o limite da área de interesse [Bezerra, Cecin e Geyer 2008].

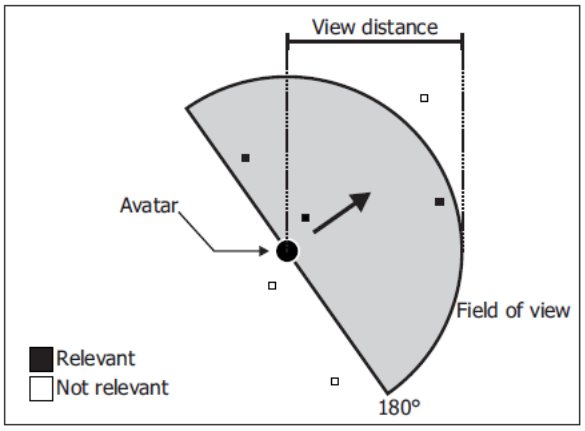

Figura 5 – Área de Interesse baseada em campo de visão. Extraído de [Bezerra, Cecin e Geyer 2008]

Área de interesse gradual: a abordagem se baseada no princípio de que os elementos mais próximos da entidade devem ser considerados mais relevantes que os elementos mais distantes, mesmo dentro da área de interesse. Desta forma, a entidade deve receber informação sobre os elementos mais relevantes com mais freqüência do que recebe dos menos relevantes [ibid.].

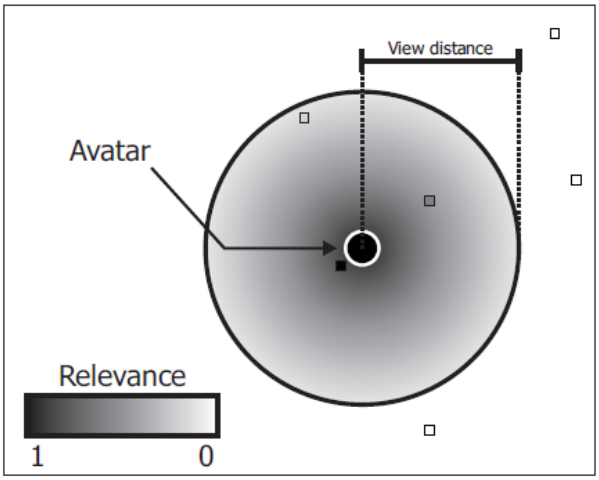

Figura 6 – Área de Interesse gradual. Extraído de [Bezerra, Cecin e Geyer 2008]

Esferas de Influência: Com o intuito de solucionar o problema do gerenciamento dinâmico de interesses em sistemas multiagentes, foi criado em 2001 por Brian Logan e Georgios Theodoropoulos um conceito chamado de Esferas de Influência. Este conceito é muito similar ao de área de interesses, com o diferencial de ser multidimensional, considerando não apenas o posicionamento do agente.

O modelo da simulação é divido em conjuntos de Processos Lógicos (LP na sigla em inglês) concorrentes, cada um mantendo uma parte disjunta do espaço do sistema. Os LPs podem ser processos lógicos de agentes ou processos lógicos de ambiente. Todos os LPs de uma simulação geram um número limitado de tipos de eventos. Diferentes tipos de eventos geram diferentes efeitos sobre o estado da simulação [Logan e Theodoropoulos 2001]. O estado da simulação é mantido através de variáveis de estado que são compartilhadas entre os agentes.

Segundo os autores, a esfera de influência de um evento pode ser entendida como “o conjunto de variáveis lidas ou atualizadas em decorrência do evento”. A esfera de influência de um LP em um dado intervalo de tempo é a união das esferas de influência dos eventos gerados por esse LP no intervalo. A interseção das esferas de influência dos LPs gera uma ordenação parcial das variáveis de estado, de forma que os primeiros elementos são os acessados pela maior quantidade de LPs e os últimos são acessados pela menor quantidade [ibid.]. Esta ordenação parcial pode ser utilizada para decompor o estado da simulação de forma a manter um determinado agente no mesmo nó da plataforma

distribuída em que se encontram os elementos com os quais ele mais interage na simulação. Isto garante uma redução no custo de comunicação entre as máquinas.

Segundo Logan e Theodoropoulos (2001), o custo computacional de calcular as esferas de influência é muito alto. Qualquer implementação eficiente pode apenas aproximar o resultado ideal.

Em [Logan e Theodoropoulos, 2001] o estado da simulação pode ser decomposto usando um novo conjunto de processos lógicos chamados *Communication Logical Processes* (CLP). Cada CLP armazenará um subconjunto do estado da simulação. Inicialmente, toda a simulação é de responsabilidade de um único CLP. Com o progresso da simulação, o CLP realiza o cálculo aproximado das esferas de influência dos agentes. Se o CLP ficar sobrecarregado (atingir um determinado limite de tráfego de rede, por exemplo), um novo CLP é criado e um subconjunto disjunto do estado da simulação é atribuído a ele, normalmente os últimos elementos da ordenação parcial gerada pela interseção das esferas de influência dos LPs. O processo é então repetido para o novo CLP, monitorando a carga e criando novos CLPs caso necessário. Esse comportamento leva a uma estrutura de árvore em que cada CLP possui como pai o CLP que requisitou sua criação. A raiz da árvore é, naturalmente, o CLP que iniciou o processo.

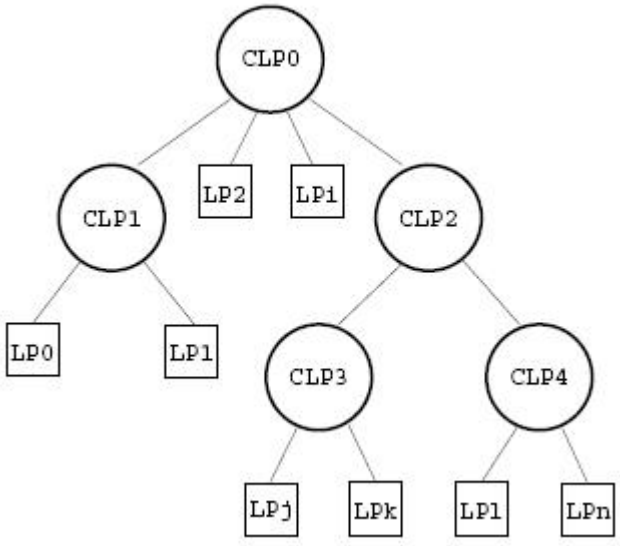

Figura 7 – Árvore de CLPs, extraído de [Logan e Theodoropoulos, 2001]

A figura 7 apresenta um exemplo de árvore de CLPs. De acordo com Logan e Theodoropoulos (2001), as partes do estado mais acessadas por um agente

tendem a ser mantidas no mesmo CLP ou, pelo menos, no CLP mais próximo possível na árvore, reduzindo o custo do roteamento de mensagens. Com o passar do tempo, novos eventos são gerados e a ordenação parcial das variáveis de estado pode sofrer alterações. A configuração da árvore deve acompanhar as mudanças nesta ordenação parcial a fim de manter o balanceamento de carga. Há mais de uma estratégia possível para manter a árvore atualizada, porém, o custo computacional da atualização deve ser considerado.

A abordagem apresentada por Logan e Theodoropoulos (2001) foi implementada em um framework chamado PDES-MAS [Oguara et al. , 2005].

3
A Abordagem Proposta

Muitas plataformas de sistemas multiagentes podem ser distribuídas entre máquinas de uma rede local ou até mesmo pela Internet. Em geral, a plataforma mantém informações sobre todos os nós conectados e é capaz de gerenciar a comunicação entre eles.

Existe troca de mensagens entre nós da rede toda vez que a esfera de influência de um agente A pertencente a um nó N contém elementos ou outros agentes pertencentes a um nó da rede diferente de N [Wang et al. 2009].

O presente trabalho propõe utilizar o conceito de esferas de influência para decompor o estado de uma simulação e distribuí-la entre os nós de uma plataforma que forneça recursos de distribuição de forma que um agente esteja no mesmo nó da rede em que se encontram os elementos com os quais ele mais interage. Diminuindo a interação dos agentes com elementos pertencentes a outros nós da rede, o custo de comunicação da simulação é reduzido e o balanceamento de carga é melhorado.

É importante destacar que cada plataforma de sistemas multiagentes possui seus próprios algoritmos de comunicação entre seus nós componentes.

3.1. Agrupando os Agentes

Em [Logan e Theodoropoulos 2001] a decomposição do ambiente é feita através da interseção das esferas de influência dos agentes. Como anteriormente comentado, essa interseção gera uma ordenação parcial das variáveis de estado, segundo a modelagem do PDES-MAS, mantendo as variáveis mais acessadas no topo da lista e as menos acessadas no final. Essa ordenação é utilizada para dividir o estado compartilhado da simulação. Neste trabalho, não foi usada a modelagem do PDES-MAS em que o estado da simulação é gerenciado através de variáveis compartilhadas. Foi aproveitado apenas o conceito de esferas de influência. Desta forma, a árvore de CLPs para armazenamento de variáveis não foi utilizada e o método de decomposição do ambiente foi alterado.

O presente trabalho propõe um agrupamento de agentes que utiliza a interseção das esferas de influência de uma maneira diferente.

Para realizar o agrupamento de forma a manter os agentes no mesmo nó da rede em que se encontram os elementos com os quais eles mais interagem, é utilizado o seguinte algoritmo:

As esferas de influência de todos os agentes do nó sobrecarregado são computadas.

O agente com a maior esfera de influência é selecionado e eleito cabeça de um grupo. A esfera de influência do agente cabeça passa a ser a esfera de influência do grupo.

Para cada outro agente no nó sobrecarregado, é calculado o percentual de interseção com a esfera de influência do grupo. Se este percentual de interseção atingir um determinado valor, o agente avaliado e sua esfera de influência são incorporados à esfera de influência do grupo e deixam de participar das próximas iterações para formações de outros grupos.

Após a avaliação de todos os agentes, um grupo fica determinado.

A partir disto, é então selecionado o agente com maior esfera de influência fora do grupo anteriormente determinado e o processo se repete até que não haja mais agentes que não façam parte de um grupo.

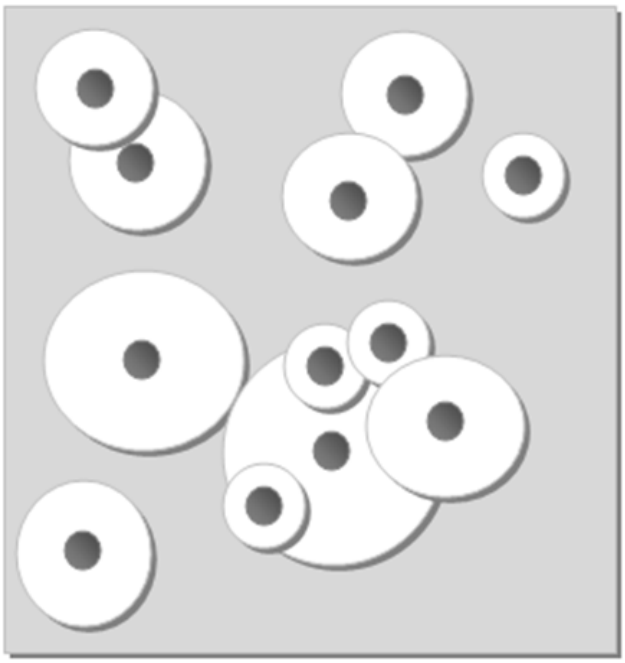

Figura 8 – Agrupamento de agentes pela interseção de suas esferas de influência

A figura 8 ilustra o agrupamento de agentes. Os círculos escuros representam agentes e os círculos claros em torno dos agentes representam suas esferas de influência. Agentes que possuem alto grau de interseção entre suas esferas de influência devem ser agrupados.

3.2. Algoritmo de particionamento

Embora a abordagem baseada em esferas de influência utilizada no framework PDES-MAS tenha apresentado resultados relevantes como técnica de gerenciamento dinâmico de interesses [Oguara et al., 2005], não é eficiente para alcançar balanceamento de carga se utilizada como único recurso. A técnica propõe iniciar a simulação utilizando um único CLP e adicionar novos CLPs cada vez que algum CLP se tornar um gargalo. Desta forma, o balanceamento de carga só é alcançado em um momento mais avançado da simulação. A vantagem de utilizar uma plataforma distribuída é parcialmente desperdiçada.

Para evitar este problema, Wang (2012) propõe a incorporação de uma fase inicial de distribuição baseada em uma técnica de balanceamento de carga estático.

Desta forma, o algoritmo proposto pode ser dividido em duas fases. Na primeira fase, de inicialização, é aplicado um método simples de balanceamento de carga estático. Na segunda fase, de execução da simulação, a carga do sistema é monitorada e atualizada dinamicamente.

Uma descrição de cada uma das fases é apresentada a seguir:

Primeira fase: A abordagem mais ingênua para particionamento de uma simulação consiste em distribuir seu estado uniformemente entre os nós da plataforma. Este será o método utilizado na primeira fase do algoritmo apresentado. Esta fase é importante para aproveitar as vantagens de possuir uma plataforma distribuída durante toda a simulação.

O algoritmo do presente trabalho exige a definição de um limite para o número de nós da rede utilizados na distribuição nesta etapa. Se o número de nós da plataforma for elevado e todos os nós forem utilizados na distribuição inicial, o custo de comunicação pode superar o ganho em poder de processamento. Este

limite pode variar de acordo com o tamanho da simulação e com a quantidade de recursos que ela demanda.

Segunda fase: Com a finalidade de evitar gargalos na simulação, são estabelecidos limites que, uma vez atingidos, classificam um nó da plataforma como sobrecarregado. Alguns critérios úteis são: uso de CPU, uso de memória e tráfego de rede.

A simulação é iniciada nos nós da plataforma selecionados na primeira fase e quando um dos limites estabelecidos é alcançado, o processo de distribuição é disparado. Os agentes do nó sobrecarregado são então agrupados aos elementos de ambiente com os quais mais interagem através da interseção de suas esferas de influência. Estes agrupamentos são migrados do nó em questão para um nó disponível na plataforma. Cada grupo é migrado de uma vez. Se o nó receptor ficar sobrecarregado, a migração continua para o próximo nó disponível. O processo é interrompido quando o nó que o disparou não está mais sobrecarregado, evitando assim migrações desnecessárias, ou quando não há mais máquinas disponíveis na plataforma.

Durante a segunda fase, a carga do sistema é monitorada constantemente. As interseções das esferas de influência dos agentes são computadas apenas quando um limite é atingido.

É importante ressaltar que a segunda fase é realizada separadamente em cada nó selecionado na fase anterior e em cada nó adicionado posteriormente ao conjunto de máquinas da distribuição. O objetivo é eliminar a necessidade de uma entidade central computando o agrupamento dos agentes de toda a simulação e efetuando a migração destes agentes. Uma entidade central deste tipo poderia se tornar um gargalo para a simulação, sendo um fator limitador que geraria problemas de escalabilidade.

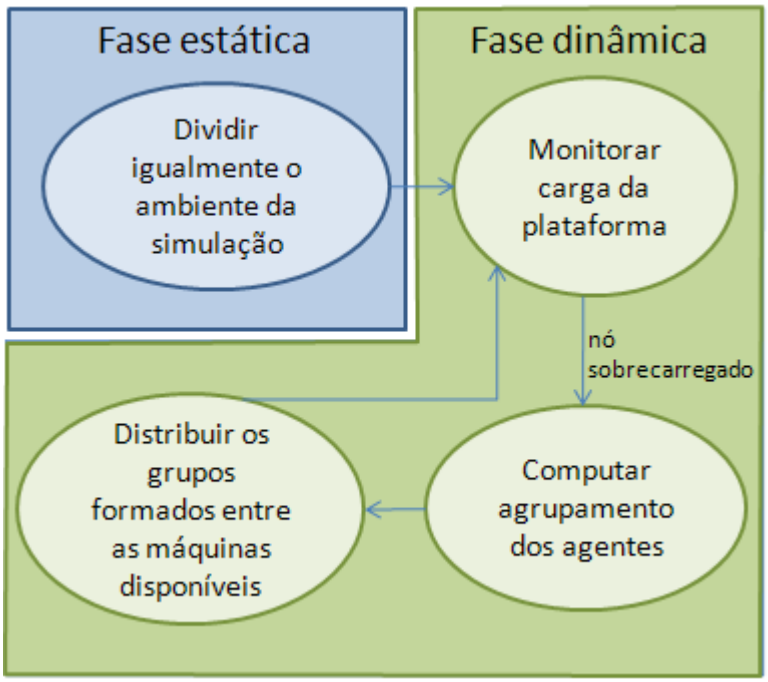

Figura 9 – Resumo do algoritmo de distribuição

A figura 9 ilustra o algoritmo proposto destacando suas duas fases. A fase estática, em azul, que ocorre apenas uma vez durante a inicialização da simulação. A fase dinâmica, em verde, se repete durante todo o tempo em que a simulação está ativa.

4
O Estudo de Caso

Para ilustrar o método proposto, foi desenvolvida uma simulação simples representando um cenário de derramamento de petróleo. Este tipo de desastre é comum e pode causar diversos tipos de impactos ao meio-ambiente. Entre os mais freqüentes é possível listar:

- envenenamento de peixes
- bloqueio da luz solar impedindo a fotossíntese das algas
- morte por afogamento de pássaros que ficaram incapazes de voar porque tiveram suas penas cobertas de petróleo.

Figura 10 – Derramamento de Petróleo

4.1. O modelo

O modelo da simulação é composto por um pequeno conjunto de classes de agentes. Há uma classe para representar os barcos recolhedores, uma classe para manchas de petróleo e uma classe representando células de mar. O ambiente é apresentado em um *grid* composto por células de mar.

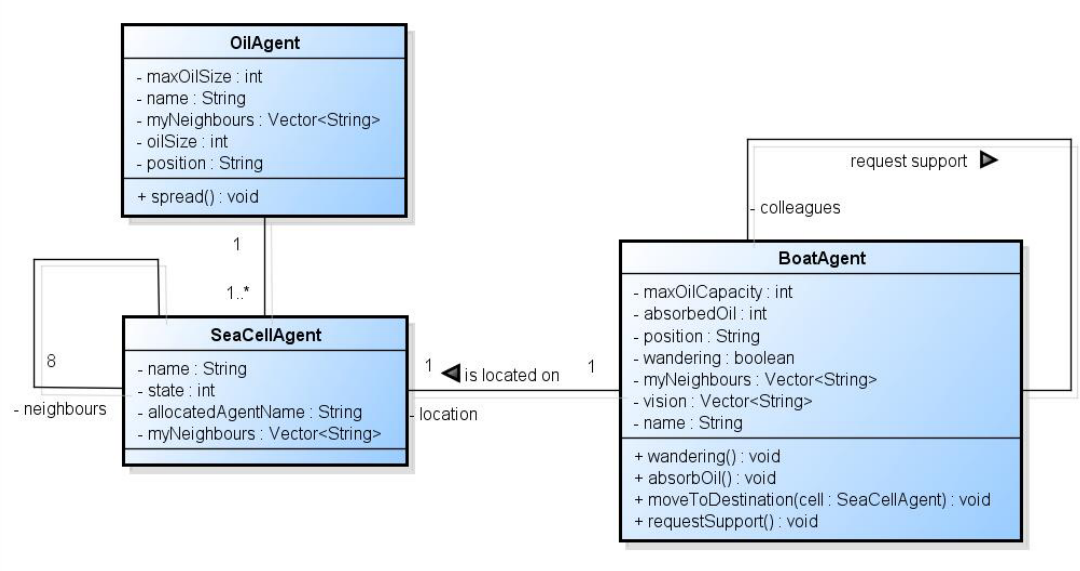

Figura 11 – Diagrama de classes da simulação

A figura 11 apresenta o diagrama de classes da simulação. Abaixo segue uma descrição de cada classe:

SeaCellAgent: é o agente que representa uma porção de mar. Há um agente deste tipo em cada célula do *grid*.

OilAgent: é o agente que representa uma mancha de petróleo. Possui apenas o comportamento de se espalhar pelo *grid*.

BoatAgent: O agente barco tem o comportamento de vaguear pelo ambiente até que uma mancha de petróleo entre em seu campo de visão. Ao avistar uma mancha, o agente barco percorre o caminho até ela e a absorve assim que a alcança. Se a capacidade do barco não for suficiente para absorver todo o petróleo, outro barco recolhedor é solicitado.

Adaptando o modelo descrito acima para incluir as classes necessárias para a aplicação do algoritmo proposto, temos o seguinte diagrama:

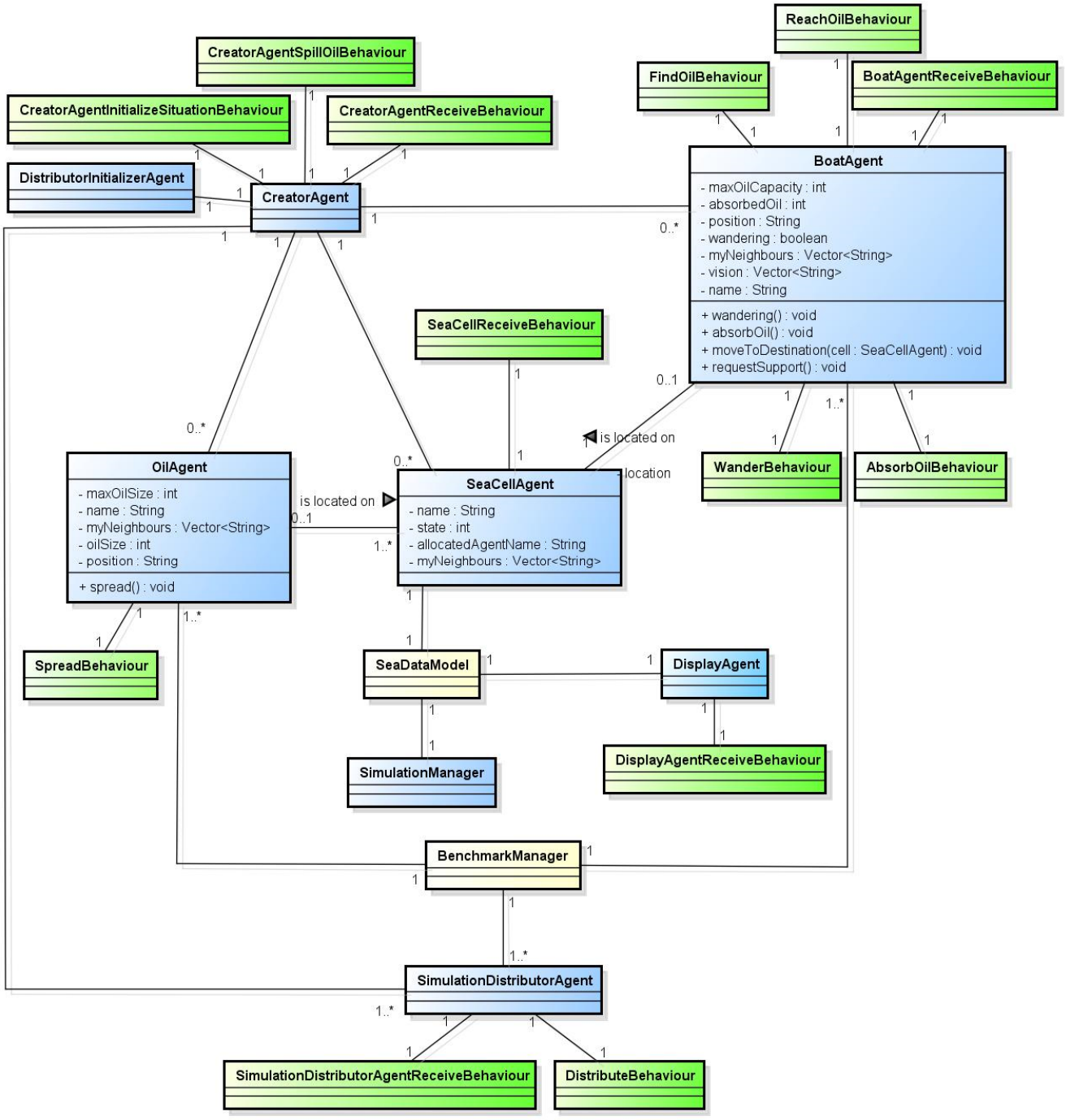

Figura 12 – Diagrama de classes da solução implementada

A figura 12 exibe o diagrama de classes anterior incorporando o núcleo do método de distribuição, ou seja, os agentes responsáveis pelo algoritmo de particionamento proposto. Encontram-se também no diagrama algumas classes relacionadas ao funcionamento geral da simulação.

Abaixo segue a descrição das classes adicionadas:

SeaDataModel: esta classe representa o *grid* composto pelas células de mar (*SeaCellAgent*). Relacionada à classe *SeaDataModel* encontram-se as classes *SimulationManager* e *DisplayAgent*.

SimulationManager: é responsável por efetuar no modelo de dados as mudanças decorrentes dos eventos gerados pelos agentes.

DisplayAgent: é o encarregado de exibir o *grid* na interface gráfica e atualizar na tela todas as mudanças realizadas no modelo de dados.

BenchmarkManager**:** é a classe responsável por armazenar os dados de acesso dos agentes e permitir o cálculo das esferas de influência. Esta classe possui métodos de apresentação de estatísticas baseadas nos dados armazenados.

CreatorAgent: o agente criador é o responsável por inserir na simulação todos os agentes do modelo original (figura 3) e por criar os agentes distribuidores em cada nó da plataforma. Inicialmente, atribui um agente de célula do mar para cada célula do *grid*. Quando a simulação é iniciada, o agente criador insere no ambiente um determinado número de manchas de óleo e de barcos. Este agente possui também a função de manter a simulação ativa, criando de tempos em tempos novas manchas de óleo para serem absorvidas pelos barcos. Quando os barcos na simulação não são capazes de absorver todo o petróleo, eles solicitam o envio de um novo barco. Isto significa enviar uma mensagem para o agente criador, que incluirá um novo agente no ambiente.

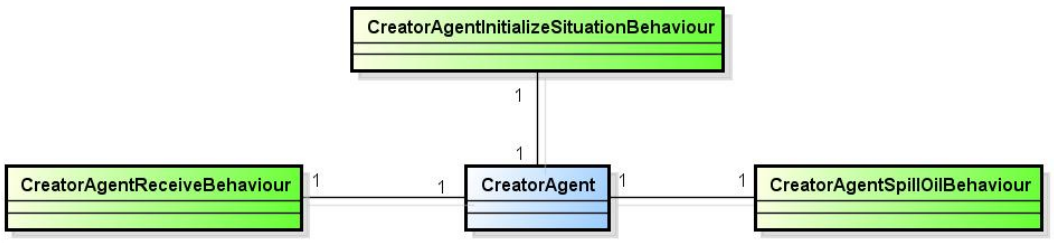

Figura 13 – CreatorAgent

SimulationDistributorAgent (SDA)**:** é o agente responsável pela distribuição dos demais agentes na plataforma. Cada nó da plataforma possui seu próprio agente distribuidor. Sempre que uma mancha de óleo ou um barco interage com o

ambiente, uma ocorrência é registrada indicando que parte do estado foi acessada. Esta informação é posteriormente utilizada pelo SDA para computar as esferas de influência dos agentes. O SDA monitora a plataforma verificando se algum limite estabelecido (uso de processamento ou uso de memória) foi alcançado. Em caso positivo, o SDA inicia o processo de distribuição de acordo com o algoritmo descrito no capítulo três.

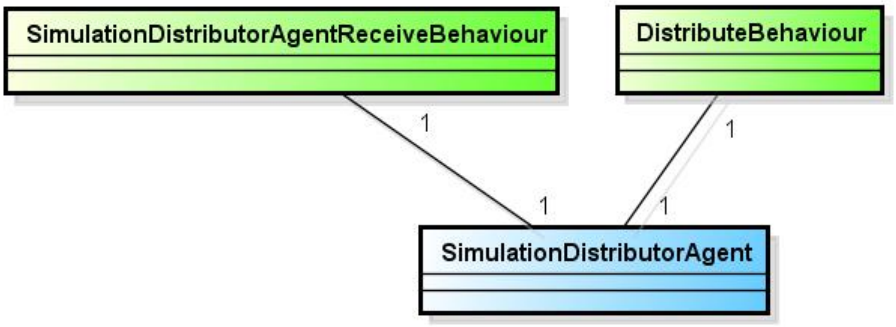

Figura 14 – SimulationDistributorAgent

DistributorInitializerAgent: é o agente responsável por monitorar a inclusão de nós na plataforma. Toda vez que um novo nó se conecta à plataforma, o *DistributorInitializerAgent* solicita ao *CreatorAgent* a criação de um *SimulationDistributorAgent* nesta nova máquina.

Como pode ser observado, o núcleo da distribuição é composto por apenas três agentes: *DistributorInitializerAgent*, *CreatorAgent* e S*imulationDistributorAgent*.

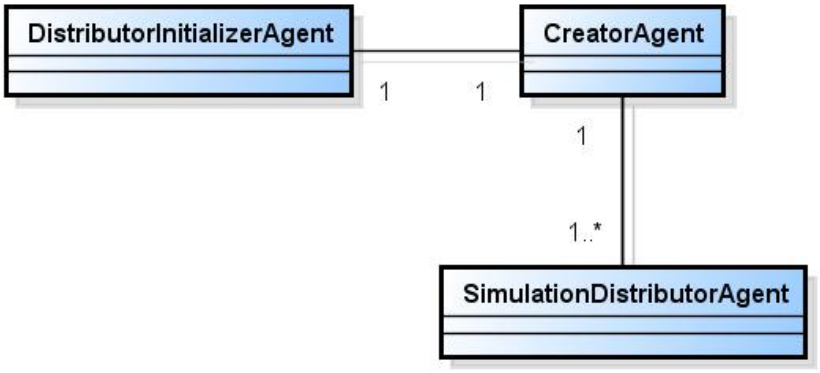

Figura 15 – Núcleo da distribuição

4.2. A simulação desenvolvida

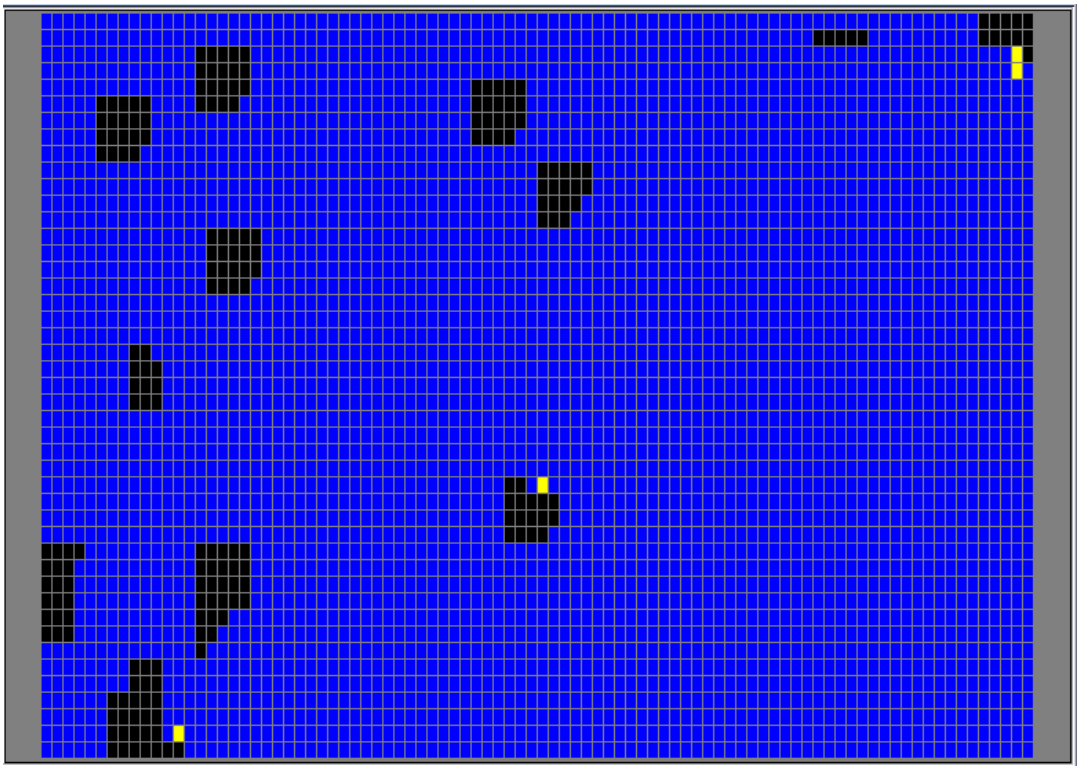

Figura 16 – Execução da simulação desenvolvida

A figura 16 apresenta a simulação desenvolvida. As células azuis do *grid* representam as células do mar. As células em preto, por sua vez, são representações das manchas de petróleo. Por último, as células em amarelo são os barcos recolhedores.

O estudo de caso implementado para o presente trabalho utilizou dois frameworks auxiliares em seu projeto: JADE framework e Agent.GUI.

4.2.1. JADE Framework

JADE é uma sigla para Java Agent DEvelopment Framework. Trata-se de um *framework* desenvolvido em Java que fornece recursos para implementação de sistemas multiagentes. O framework JADE foi desenvolvido de acordo com o padrão FIPA [Bellifenime et al., 2005]. Sua plataforma de agentes foi utilizada no projeto desenvolvido no presente trabalho porque possui todos os recursos de distribuição necessários, além de possuir uma ampla e ativa comunidade de desenvolvedores.

A plataforma do *framework* JADE é composta por um conjunto de *Agent Containers*, local onde os agentes residem e que provê o suporte para execução dos mesmos. Além da possibilidade de adicionar à plataforma contêineres de outras máquinas da rede, é possível mover os agentes entre estes contêineres. Quando um contêiner remoto é criado, ele pode se juntar à plataforma em tempo de execução.

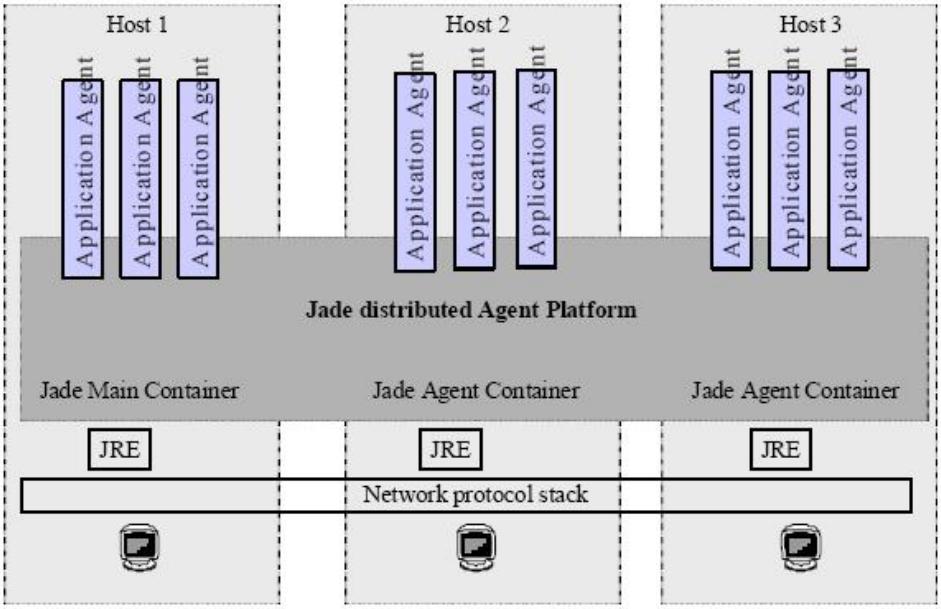

Figura 17 – Plataforma distribuída do framework JADE. Extraída de [Bellifenime et al., 2012]

A figura 17 ilustra a plataforma JADE composta de contêineres alocados em diferentes *hosts*.

Essa característica da plataforma JADE foi um dos principais fatores para adoção do *framework* no estudo de caso do presente trabalho. Cada máquina que se conecta à plataforma mantém um contêiner em que os agentes residirão.

Para compreender melhor alguns recursos oferecidos pela plataforma JADE, abaixo segue uma descrição do modelo básico de uma plataforma de sistemas multiagentes segundo o padrão FIPA.

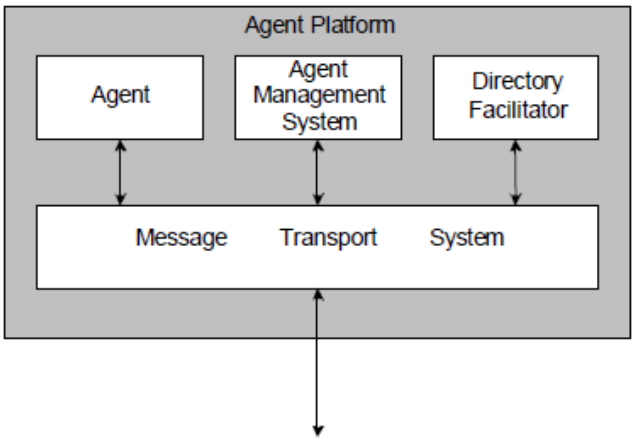

Figura 18 – Plataforma distribuída do padrão FIPA. Extraída de [Bellifenime et al., 2012]

O *Agent Management System (AMS)* é um agente responsável por controlar o acesso e o uso da plataforma. Existe apenas um AMS por plataforma. O AMS mantém um diretório de identificadores de agentes (AID) e de estados de agentes. Todo agente deve ser registrado no AMS para obter um AID válido. Todas as informações referentes à plataforma devem ser solicitadas ao AMS. Como exemplos do tipo de consulta feita a este agente podem ser citados: número de contêineres conectados à plataforma; nome destes contêineres; número de agentes em um determinado contêiner ou em toda a plataforma; identificação destes agentes, entre outros.

O *Directory Facilitator (DF)* é o agente que provê o serviço padrão de páginas amarelas na plataforma.

O sistema de transporte de mensagens, conhecido como *Agent Communication Channel (ACC)*, é o responsável por toda troca de mensagens na plataforma, inclusive as mensagens entre contêineres remotos.

O *framework* JADE atende aos requisitos do padrão FIPA. O AMS e o DF são inicializados no contêiner principal da plataforma assim que esta é criada e o ACC permanece ativo todo o tempo [ibid.].

Na solução proposta o *DistributorInitializerAgent* possui um comportamento chamado *AMSListenerBehaviour* que monitora o AMS para identificar quando um novo contêiner é adicionado à plataforma. Nestes casos, o *DistributorInitializerAgent* envia uma mensagem ACL ao *CreatorAgent* para que ele crie um *DistributorAgent* no novo nó da plataforma.

4.2.1.1. Comunicação entre os agentes

A comunicação entre os agentes é efetuada através de mensagens que utilizam uma linguagem específica definida no padrão FIPA. A linguagem em questão é chamada *Agent Communication Language (ACL)*. Os agentes são capazes de enviar e receber mensagens ACL. As mensagens ACL possuem uma série de parâmetros que devem ser preenchidos de acordo com seu fim. O padrão FIPA define o conjunto básico de parâmetros para mensagens ACL mas o usuário pode criar seus próprios parâmetros caso deseje. Abaixo segue a lista de parâmetros que constam no padrão FIPA.

Parâmetro	Categoria de Parâmetros
performative	Tipo de ato comunicativo
sender	Participante na comunicação
receiver	Participante na comunicação
reply-to	Participante na comunicação
content	Conteúdo da mensagem
language	Descrição do Conteúdo
encoding	Descrição do Conteúdo
ontology	Descrição do Conteúdo
protocol	Controle da Conversa
conversation-id	Controle da Conversa
reply-with	Controle da Conversa
in-reply-to	Controle da Conversa
reply-by	Controle da Conversa

Tabela 1 – Tabela de parâmetros de uma mensagem ACL

O *framework* JADE provê o transporte de mensagens ACL entre agentes que se encontram na mesma plataforma de maneira eficiente [ibid.].

4.2.1.2. Comportamentos dos agentes

As ações dos agentes são modeladas através de comportamentos. No *framework* JADE um comportamento é um objeto da classe *Behaviour*. Comportamentos podem ser adicionados em qualquer momento da execução de um agente, não apenas em sua inicialização. Existem diversos tipos de

comportamentos definidos. Abaixo segue uma tabela com uma breve descrição de cada um:

Tipo de Comportamento	Nome	Descrição
Comportamentos Simples	OneShotBehaviour	Comportamentos que devem ser executados apenas uma vez.
	CyclicBehaviour	Comportamentos que devem ser executados eternamente.
Comportamentos Compostos	SequentialBehaviour	Possui uma lista de sub-comportamentos que são executados seqüencialmente.
	ParallelBehaviour	Possui uma lista de sub-comportamentos que são executados concorrentemente.
	FSMBehaviour	Possui uma lista de sub-comportamentos que são executados como uma Máquina de Estados Finita.
Comportamentos Especiais	WakerBehaviour	Comportamentos que devem ser executados apenas uma vez logo após aguardar um determinado período de tempo.
	TickerBehaviour	Comportamento que executa uma tarefa periodicamente em intervalos de tempo constante.

Tabela 2 – Comportamentos de agentes do JADE *Framework*

4.2.2. Agent.GUI

Para exibir graficamente a interação entre os agentes foi utilizado o *framework* Agent. GUI. O *framework* provê uma interface gráfica pré-definida que pode ser adaptada às necessidades do desenvolvedor. É completamente desenvolvido em Java e baseado no *framework* JADE [Derksen, Branki e Unland 2011].

O *framework* Agent.GUI fornece um sistema que facilita a distribuição de uma simulação baseada em agentes. Este sistema é chamado de *Background System* e não será descrito detalhadamente porque o presente trabalho não faz uso desta funcionalidade.

O projeto implementado no trabalho proposto utilizou o *framework* Agent.GUI porque ele fornece um modelo de ambiente simples de utilizar e permite a visualização deste ambiente em um *grid*.

Outra funcionalidade relevante para o presente trabalho é o monitoramento de carga da plataforma. O Agent.GUI possui um serviço chamado *LoadService* que monitora constantemente a carga da plataforma, inclusive quando a plataformas é distribuída. Este serviço é utilizado no presente trabalho para identificar quando iniciar e quando interromper processos de distribuição em cada um dos nós da plataforma. Ele provê informações como: percentual de utilização de capacidade de processamento, percentual de utilização de memória, entre outras.

O *framework* também oferece uma interface de controle da simulação que permite diversos níveis de parametrização.

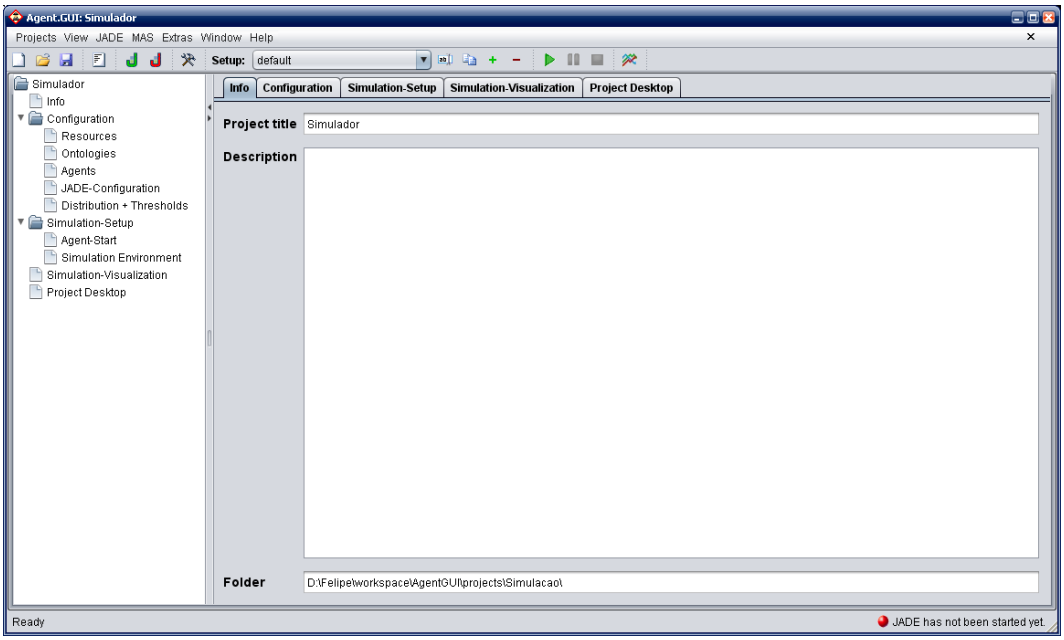

Figura 19 – Interface de controle do Agent.GUI

A interface de controle exibida na figura 19 pode ser personalizada e permite ao desenvolvedor adicionar suas próprias funcionalidades sem dificuldade. O *framework* fornece uma classe chamada *PlugIn* que pode ser estendida para incluir elementos na interface principal da aplicação.

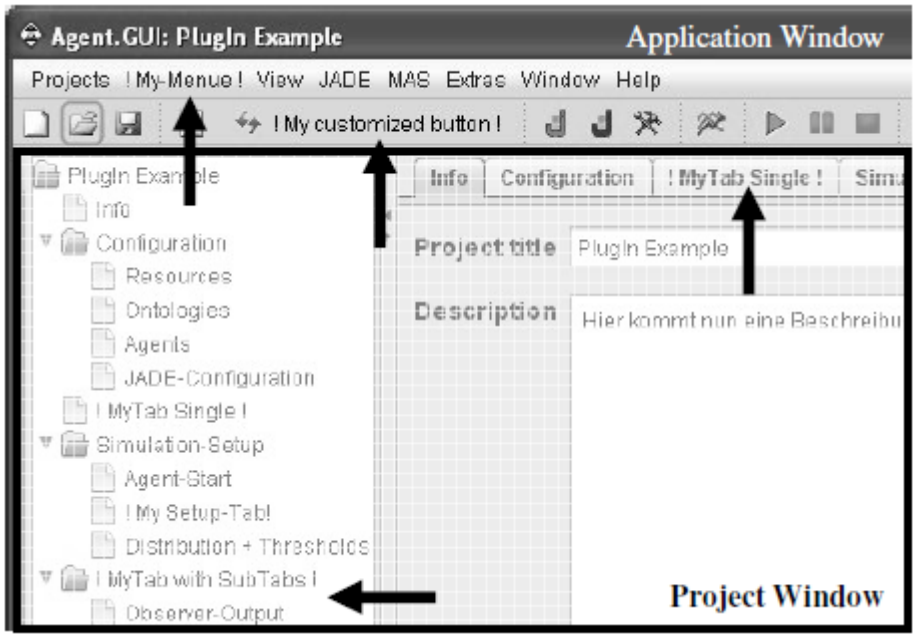

Figura 20 – Interface de controle do Agent.GUI estendida. Extraído de [Derksen, Branki e Unland 2011]

Conforme pode ser visto na figura 20 é possível adicionar botões, abas e itens de menu à interface principal do Agent.GUI.

4.3. Experimentos

Para verificar a viabilidade da solução proposta foram executados dois experimentos, utilizando o estudo de caso, que ilustram como o desempenho da simulação desenvolvida é afetado por cada uma das duas fases do algoritmo proposto. Além disto, foi realizado um terceiro experimento para estimar a tendência de comportamento do desempenho da solução ao escalar novas máquinas.

Nos dois primeiros experimentos foram utilizados dois computadores AMD Sempron LE-1250 de 2.21 GHz e 2GB de memória RAM para executar a simulação. Em cada um dos experimentos a simulação foi executada de três maneiras diferentes. Primeiramente, a simulação contou apenas com a fase estática do algoritmo proposto. Logo após, a simulação utilizou somente a fase dinâmica da distribuição. Em seguida, as duas fases foram incluídas no processo. Cada uma das três variações foi executada cinco vezes e os resultados apresentados são médias destas execuções.

No terceiro experimento foram utilizados três computadores AMD Sempron LE-1250 de 2.21 GHz e 2GB de memória RAM. Neste caso, as duas fases da solução proposta foram adotadas. A simulação foi executada inicialmente utilizando apenas um computador. Em seguida, foram utilizados dois computadores e, por fim, três computadores em conjunto.

Em todos os casos, a simulação durou cento e vinte segundos. O cenário inicia com quinze barcos e quinze manchas de petróleo e a cada cinco segundos, uma nova mancha de petróleo aparece no mar.

4.3.1. Primeiro Cenário

No primeiro cenário foi utilizado um *grid* composto por vinte linhas e quarenta colunas, ou seja, oitocentas células de mar.

Grid 20x40 (800 células de mar) – 120 segundos	
Fase(s) Utilizada(s)	Número de eventos por segundo
Apenas 1ª fase	356
Apenas 2ª fase	901
Ambas as fases	855

Tabela 3 – Resultados do primeiro experimento

A tabela 3 exibe os resultados do primeiro experimento. A execução que utiliza apenas a primeira fase apresentou, como esperado, resultado bastante inferior às outras. Vale lembrar que na primeira fase, apenas o ambiente da simulação é dividido entre às máquinas.

A tabela também apresenta um resultado superior para a execução que conta apenas com a segunda fase em relação à que conta com as duas fases. Como neste cenário de apenas 800 células o ambiente é reduzido, as máquinas não ficam sobrecarregadas facilmente. A utilização da primeira fase faz com que parte do ambiente comece a simulação em um servidor diferente do servidor em que estão os agentes barcos e manchas. Isto inclui um custo de comunicação no início da simulação que supera o benefício da utilização da primeira fase, visto que em um ambiente pequeno a carga fica balanceada rapidamente.

4.3.2. Segundo Cenário

No segundo cenário, o *grid* utilizado era composto por quarenta e cinco linhas e noventa colunas, totalizando quatro mil e cinqüenta células de mar.

Grid 45x90 (4050 células de mar) – 120 segundos	
Fases Utilizadas	Número de eventos por segundo
Apenas 1ª fase	150
Apenas 2ª fase	450
Ambas as fases	622

Tabela 4 – Resultados do segundo experimento

É possível perceber pelos números apresentados na tabela 4 que neste cenário a utilização das duas fases do algoritmo proposto em conjunto foi mais eficiente que a utilização da segunda fase apenas. Como o tamanho do *grid* é consideravelmente maior em relação ao experimento anterior, as máquinas passaram a ficar sobrecarregadas com mais facilidade. Neste caso, eliminar a primeira fase da distribuição significa levar mais tempo para alcançar o balanceamento de carga.

Outro ponto que pode ser observado é que o número de eventos por segundo diminui bastante no segundo experimento em relação ao primeiro. O motivo é o aumento na necessidade de poder de processamento devido ao tamanho do *grid*.

4.3.3.Terceiro Cenário

No terceiro cenário, quarenta e cinco linhas e noventa colunas compunham o *grid*, totalizando quatro mil e cinquenta células de mar.

***Grid* 45x90 (4050 células de mar) – 120 segundos**	
Máquinas utilizadas	Número de eventos por segundo
1	152
2	619
3	991

Tabela 5 – Resultados do terceiro experimento

Como pode ser visto na tabela 5, o resultado obtido utilizando 3 máquinas é muito superior a simplesmente triplicar o resultado da execução com apenas uma máquina. No entanto, o ganho da inclusão de uma terceira máquina não foi tão grande quanto o da inclusão do segundo computador. A adição de mais nós na rede aumenta o custo de comunicação, gerando impacto no desempenho.

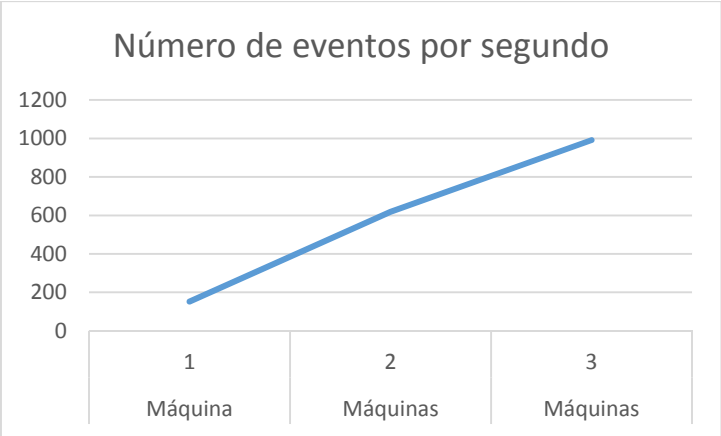

Figura 21 – Curva de progressão do número de eventos por segundo de acordo com o número de máquinas

O resultado ilustra a tendência de que a progressão da curva de eventos por segundo seja mais suave com o aumento do número de computadores.

5 Trabalhos Relacionados

Este capítulo apresenta conjuntos de trabalhos que possuem temas relacionados ao trabalho proposto ou que serviram de base e inspiração para o mesmo. A seguir estes trabalhos serão comentados e analisados sob a visão do trabalho proposto.

5.1. Particionamento de simulações de multidões baseadas em agentes

Em [Wang et. al. 2009] os autores investigam a aplicação de um algoritmo de particionamento de simulações de multidão baseado em *clustering*. *Clustering* é uma técnica usada para agrupar objetos ou valores similares.

Dentre os trabalhos relacionados, este é o que mais se aproxima do objetivo do presente trabalho. Assim como no trabalho proposto, os autores buscam fornecer uma solução para agrupar os agentes de uma simulação de forma a manter o mínimo possível de comunicação entre os processadores da plataforma. O artigo busca minimizar a ocorrência de sobreposição de áreas de interesses de agentes que se encontrem em servidores distintos. Desta forma, reduz-se a troca de mensagens entre os computadores.

Os autores consideram que agentes com áreas de interesses sobrepostas devem ser agrupados juntos. Além disto, consideram que agentes próximos que possuem o mesmo comportamento tendem a interagir agora e no futuro. Propõe-se utilizar, além da informação de posicionamento de um agente, seu estado e sua velocidade para predizer agrupamentos de agentes.

Em [Wang, Lees e Cai 2012] é utilizado um algoritmo muito similar ao apresentado em [Wang et. al. 2009]. A principal diferença é que além de utilizar informações como estado e velocidade do agente para agrupar as entidades, o método utiliza informações sobre movimentos futuros do agente (através de suas metas). Em situações específicas como cruzamento de duas multidões, a nova técnica apresenta vantagens em relação ao trabalho anterior. Outra diferença é que o método utilizado para agrupar os agentes é um método baseado em *grid*. O

ambiente da simulação é dividido em um *grid* em que todas as células possuem o mesmo tamanho. O algoritmo de agrupamento de agentes é executado a partir da primeira célula do *grid*. Se a variância de metas em relação à posição em que os agentes da célula devem atingir for baixa, ou seja, a maioria dos agentes possui o mesmo destino ou destinos próximos, os elementos da célula são agrupados. Em seguida, o processo ocorre novamente em cada uma das demais células, sempre calculando a variância em relação ao grupo formado. É importante ressaltar que este método possui uma etapa de inicialização em que o ambiente é distribuído igualmente entre os computadores antes da execução da simulação. Cada computador divide sua porção do ambiente em células e executa o algoritmo de agrupamento de agentes simultaneamente aos demais.

O presente trabalho adota a idéia de uma fase de inicialização apresentada em [Wang, Lees e Cai 2012] em que o ambiente é igualmente distribuído entre os computadores, tomando apenas o cuidado de não permitir que o número de computadores utilizados nesta primeira fase seja exagerado a ponto de gerar um custo de comunicação desnecessário. Outra prática de [Wang, Lees e Cai 2012] adotada no presente trabalho é executar o algoritmo de agrupamento de agentes em cada máquina separadamente. Eliminando a necessidade de todo o processamento de cálculo de agrupamento de agentes de todos os nós da rede ser realizado por uma entidade central, evita-se a possibilidade de que a máquina responsável por esta entidade se torne um gargalo para toda a simulação.

Como [Wang, Lees e Cai 2012] é focado em simulação de multidões, é pertinente utilizar as metas dos agentes para prever agrupamentos. No entanto, em simulações em que as entidades não possuem o comportamento de formar multidões, rebanhos ou aglomerações, esta estratégia gera um *overhead* desnecessário.

5.2. Gerenciamento Dinâmico de Interesses e o balanceamento de carga em simulações distribuídas

Em [Logan e Theodoropoulos 2001], [Lees et al. 2004] e [Oguara et al. 2005] é proposta a utilização do conceito de esferas de influência para decompor o estado de uma simulação e distribuí-la através de uma estrutura de dados própria que visa melhorar o balanceamento de carga. O estado da simulação é gerenciado

através de variáveis que são distribuídas entre os computadores da rede. Os agentes são mantidos em processadores diferentes das variáveis de estado. As variáveis são mantidas em nós de uma árvore de processos lógicos de comunicação e devem estar sempre na menor distância possível dos agentes que as acessam mais freqüentemente. A estrutura é complexa de reproduzir e o custo de reorganizar a árvore para obter o melhor resultado é geralmente muito alto, sendo questionável se o ganho em desempenho compensaria o custo de tal tarefa.

O método é implementado no framework PDES-MAS apresentado mais detalhadamente em [Oguara et al. 2005].

O trabalho de Logan e Theodoropoulos (2001) foi o principal motivador do presente trabalho. Ele apresenta o conceito de esferas de influência e descreve como elas podem ser utilizadas para decompor o estado de uma simulação. O presente trabalho utiliza o conceito de esferas de influência, porém, propõe um algoritmo diferente para agrupar os agentes com os elementos com os quais eles mais interagem.

5.3. Algoritmos de particionamento eficiente

Entre os algoritmos de particionamento de simulações e ambientes virtuais distribuídos que alcançam os melhores resultados está o apresentado em [Lui e Chan 2002]. Conhecido como *Linear Optimization Technique (LOT)*, o algoritmo define uma função de qualidade chamada C_p para avaliar as partições. Esta função de qualidade possui dois parâmetros. O primeiro diz respeito à carga de trabalho gerada na simulação. Para reduzir este parâmetro é necessário dividir a carga de trabalho entre todos os computadores da rede. O segundo parâmetro é relativo ao total de mensagens trocadas entre servidores. Para reduzir este parâmetro as entidades que possuem interseções em suas áreas de interesses devem ser mantidas no mesmo servidor. Os autores propõem um algoritmo que utiliza esta função de qualidade para realocar entidades nos servidores. O algoritmo proposto por Lui e Chan (2002) é dividido em três etapas. A primeira se trata de aplicar um algoritmo chamado *Recursive Bisection Partition (RBP)*, que utiliza o conceito de divisão e conquista para definir uma partição inicial. A segunda etapa se trata de aplicar um algoritmo chamado *Layering Partitioning (LP)* para balancear a carga da simulação. A terceira etapa é referente à execução de algoritmo chamado

Communication Refinement Partitioning (CRP) para reduzir a comunicação entre os servidores.

Em [Morillo, Orduña e Fernández 2004] é apresentado um estudo que compara diversos algoritmos evolutivos de particionamento de ambientes virtuais distribuídos e o algoritmo *LOT* é utilizado como parâmetro principal da comparação devido ao seu excelente desempenho e à função de qualidade proposta por Lui e Chan (2002).

6 Conclusões e Trabalhos Futuros

Este capítulo apresenta as conclusões do trabalho desenvolvido e enumera os trabalhos que merecem receber atenção no futuro.

O presente trabalho propôs uma nova abordagem para o particionamento de simulações distribuídas baseadas em agentes. A abordagem sugere o uso de técnicas de gerenciamento de interesses para manter cada agente no mesmo nó da rede em que se encontram os elementos com os quais ele mais interage. Desta forma, é reduzido o custo de comunicação entre os computadores.

Embora haja outros trabalhos que agrupam os agentes baseados na interseção de suas esferas de influência [Logan e Theodoropoulos 2001], [Lees et al. 2004] e [Oguara et al. 2005], este trabalho propõe um novo método para esta tarefa.

O algoritmo de particionamento proposto é divido em duas fases. A primeira fase, de inicialização, se trata de uma distribuição estática da simulação em que o ambiente virtual é dividido igualmente entre os nós da rede selecionados para este fim. A segunda fase é referente à parte dinâmica do particionamento, em que cada nó da rede monitora sua carga de trabalho e efetua medidas de distribuição de seus agentes quando necessário.

Resultados de experimentos baseados no estudo de caso desenvolvido mostram que a abordagem apresenta melhorias no desempenho geral da simulação. No entanto, em simulações em que o balanceamento de carga pode ser alcançado rapidamente (simulações com poucos agentes), pode não ser indicado utilizar a primeira fase do algoritmo. Já em simulações em que o balanceamento de carga pode levar mais tempo para ser alcançado (simulações de larga escala), a utilização da primeira fase pode melhorar consideravelmente o desempenho. Vale destacar que o foco do presente trabalho é justamente o segundo caso.

6.1. Contribuições

Abordagem para particionamento dinâmico de simulações distribuídas: a principal contribuição do trabalho apresentado foi uma nova abordagem para o particionamento dinâmico de simulações distribuídas baseadas em agentes. A abordagem proposta tem sua execução dividida em duas fases: uma estática e uma dinâmica. Para agrupar os agentes de forma a reduzir o custo de comunicação é utilizada uma técnica de gerenciamento de interesses.

Estudo de caso: para verificar a efetividade da solução foi desenvolvido um estudo de caso. O cenário apresentado é uma simulação de derramamento de petróleo que contém barcos recolhedores perambulando pelo mar à procura de manchas de petróleo. Sempre que um barco encontra uma mancha, ele a recolhe. Se não tiver capacidade para isto, solicita a ajuda de outro barco.

6.2. Principais Limitações

O presente trabalho preocupa-se em fornecer um método de particionamento de simulações que mantenha reduzido o custo de comunicação entre processadores ao longo da simulação. Porém, não há compromisso com questões como sincronismo de eventos, segurança e outros problemas conhecidos próprios à distribuição de simulações.

6.3. Trabalhos Futuros

Entre os trabalhos futuros podemos citar: o desenvolvimento de uma técnica inteligente de seleção do nó da rede mais adequado para receber os agentes em uma etapa de distribuição; a criação de novos estudos de caso e a implementação de mecanismos de sincronismo de eventos.

7
Referências Bibliográficas

Bellifemine, F.; Bergenti, F.; Caire, G.; Poggi, A., JADE - A Java Agent Development Framework. **In Proceedings of Multi-Agent Programming**, p. 125-147, 2005.

Bellifemine, F.; Caire, G.; Trucco, T.; Rimassa, G. **JADE Programmer's Guide**. Available at: http://jade.tilab.com/doc/programmersguide.pdf Access: 10 dec. 2012

Bezerra, C. E; Cecin, F. R; Geyer, C. F. R. A3: a novel interest management algorithm for distributed simulations of MMOGs. **In Proceedings of the 2008 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications**, p. 35-42, 2008.

Caire, G.; Rimassa, G.; Bellifemine, F. JADE: a versatile run-time for distributed applications on mobile terminals and networks. **In Proceedings of SMC**, p. 1882-1888, 2004.

Derksen, C.; Branki, C.; Unland, R. Agent.GUI: A Multi-agent Based Simulation Framework. **In Proceedings of FedCSIS**, p. 623-630, 2011.

Ewald, R.; Chen, D.; Theodoropoulos, G. K.; Lees, M.; Logan, B.; Oguara, T.; Uhrmacher, A. M. Performance Analysis of Shared Data Access Algorithms for Distributed Simulation of Multi-Agent Systems. **In Proceedings of the 20th Workshop on Principles of Advanced and Distributed Simulation**, 2006. PADS 2006, p. 29 – 36, 2006.

Foundation for Intelligent Physical Agents FIPA. **FIPA ACL Message Structure Specification**, 2002.

Fujimoto, Richard M. **Parallel and Distributed Simulation Systems**. New York: Wiley Interscience, 2000. 300 p.

Lees, M.; Logan, B.; Minson, R.; Oguara, T.; Theodoropoulos, G. Distributed Simulation of MAS. **In Proceedings of the Joint Workshop on Multi-Agent and Multi-Agent-Based Simulation (MAMABS'04)**, p. 21-30, 2004.

Lees, M.; Logan, B.; Minson, R.; Oguara, T.; Theodoropoulos, G. Modelling Environments for Distributed Simulation. **In 1st International Workshop on Environments for Multi-Agent Systems (E4MAS), in conjunction with the 3rd International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS04)**, p. 150-167, 2004.

Lees, M.; Logan, B.; Theodoropoulos, G. 2007. **Distributed Simulation of Agent-based Systems in HLA**, ACM Transactions on Modelling and Computer Simulation, Vol. 17, 3, Available at: http://doi.acm.org/10.1145/1243991.1243992) ISSN: 1049-3301.

Li, Y.; Fujimoto, R.; Hunter, M.; Suh, W. An Interest Management Scheme for Mobile Peer-to-Peer Systems. **In Proceedings of the 2011 Winter Simulation Conference**, p. 2752-2764, 2011.

Logan, B.; Theodoropoulos, G. The distributed simulation of multi-agent systems. **In Proceedings of the IEEE 89(2)**, p. 174-186, 2001.

Lui, J.C.S.; Chan, M.F. An Efficient Partitioning Algorithm for Distributed Virtual Environment Systems**. In Proceedings of IEEE Transactions on Paralleland Distributed Systems**. p. 193 – 211, 2002

Machado, L. E.; Feijó, B.; Kozovits, L.E; Selected Techniques to Improve Communication in Massively Multiplayer Games. **In Proceedings of SBGames**, 2006.

Minson, R.; Theodoropoulos, G. "An Adaptive Interest Management Scheme for

Distributed Virtual Environments". **In Proceedings of Principles of Advanced and Distributed Simulation**, 2005. PADS 2005. Workshop, 273–281.

Miller, D.; Thorpe, J. A. SIMNET: The advent of simulator networking**, In Proceedings of the IEEE 83(8)**, IEEE, 1114 – 1123, 1995

Morgan, G.; Lu, F.; Storey, K. 2005. "Interest Management Middleware for Networked Games". **In I3D'05: Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games**, edited by A. Lastra, M. Olano, D. P. Luebke, and H. Pfister, 57–64. New York, NY, USA: ACM.

Morillo, P.; Orduña J. M.; Fernández, M. A comparison study of evolutive algorithms for solving the partitioning problem in distributed virtual environment systems. **Journal of Parallel Computing** - Special issue: Parallel and nature inspired computational paradigms and applications. Vol. 30, Issue 5-6, p. 585 610, 2004

Morse, K. L. **Interest management in large scale distributed simulations**, Tech. Rep. 96-27, Department of Information and Computer Science, University of California, Irvine, 1996.

Oguara, T.; Chen, D.; Theodoropoulos, G.; Logan, B., and Less, M. An Adaptive Load Management Mechanism for Distributed Simulation of Multi-agent Systems. **Proceedings of the Ninth IEEE International Workshop on Distributed Simulation and Real-Time Applications**, pp. 179–186, 2005.

Parunak, H. V. D.; Brueckner, S.; Fleischer, M.; Odell, J. **A Design Taxonomy of Multi-Agent Interactions**, p. 123-137 of : Giorgini, P.; Müller, J. P.; Odell, J. Agent-Oriented Software Engineering IV : 4th International Workshop, AOSE 2003, Melbourne, Australia, July 15, 2003, Revised Papers. Lecture notes in computer science LNCS, vol. 2935. Springer, 2003.

Strassburger, S.; Schulze, T.; Fujimoto, R. Future Trends in Distributed

Simulation and Distributed Virtual Environment: Results of a Peer Study. **In Proceedings of the Winter Simulation Conference**, p. 777-785, 2008.

Theodoropoulos, G.; Logan, B. An Approach to Interest Management and Dynamic Load Balancing in Distributed Simulation. **In Proceedings of the 2001 European Simulation Interoperability Workshop**, p. 565-571, 2001.

Wang, Y.; Lees, M.; Cai, W. Grid-based partitioning for large-scale distributed agent-based crowd simulation. **In Proceedings of the Winter Simulation Conference**, p. 241-241, 2012.

Wang, Y.; Lees, M.; Cai, W.; Zhou, S.; Low, M. Y. H. "Cluster Based Partitioning for Agent-Based Crowd Simulations". **In Proceedings of the 2009 Winter Simulation Conference**, p. 1047-1058, 2009.

Wooldridge, M. **An Introduction to Multiagent Systems**, Hoboken, NJ: Wiley, p. 348, 2012

Wooldridge, M.; Jennings, N.R., **Intelligent Agents: Theory and Practice**. Knowledge Engineering Review, 10(2), p. 115-152, 1995