Simulando uma peixaria com Sistema de Agentes Autônomos Jason

Olimar Teixeira Borges

[1]Escola Politécnica - Informática
Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Porto Alegre – RS – Brasil

olimar.borges@acad.pucrs.br

Resumo. *Este trabalho é o relatório final apresentado na disciplina de Agentes Autônomos, que propõe uma aplicação multiagente de uma peixaria que envolve a pesca de peixes no oceano até o seu descarregamento no distribuidor, modelada utilizando Jason e CArtAgO. A tripulação do barco e o time de carregadores são responsáveis pela pesca, carregamento e descarregamento dos peixes, durante todo o processo da aplicação. O objetivo é realizar simulações, com algumas configurações diferentes, demostrando os resultados dos comportamentos gerados ao final de cada simulação.*

1. INTRODUÇÃO

A área de Sistemas Multiagentes se interessa pelo estudo de agentes autônomos em um universo Multiagentes [Demazeau and Müller 1990]. O termo autônomo designa o fato de que os agentes têm uma existência própria, independente da existência de outros agentes. Como não existe um problema pré-definido que o sistema deve resolver, o objetivo é conceber os meios a partir dos quais pode-se assegurar que agentes desejem cooperar e efetivamente o façam, com o intuito de resolver um problema específico quando este for apresentado ao sistema [Alvares and Sichman 1997].

Para o desenvolvimento deste trabalho, foi escolhido o contexto de uma empresa de peixaria, que contemplará desde a pescaria em alto mar até o transporte dos peixes em terra. Durante o processo de navegação em alto mar, haverão barcos com capacidades variadas de armazenamento de peixes, que serão responsáveis pela pesca e descarregamento de peixes no porto e que possuirão redes com capacidades limitadas para a pesca. Os barcos iniciam no porto e cada um possui três tripulantes, sendo o capitão, o pescador e o auxiliar de pesca. O capitão será responsável pelo controle e navegação do barco e por determinar o local da pesca. Já o pescador possui responsabilidade de jogar e recolher as redes. O auxiliar de pesca será responsável por puxar e soltar a âncora do barco, como também retirar os peixes das redes.

Para o processo de transporte dos peixes, caminhões serão responsáveis por recolher os peixes deixados pelos barcos no porto e levar até o distribuidor. Os caminhões também possuirão capacidades variadas de armazenamento de peixes e cada caminhão possuirá um motorista e um carregador. O motorista será responsável pelo controle e direção do caminhão, verificando a rota até o distribuidor pelo GPS. Já o carregador, possui como responsabilidades localizar o carregamento de peixes no porto e carregar o caminhão.

A programação dos agentes foi realizada em Jason [Bordini et al. 2007] que é uma extensão da linguagem de programação abstrata chamada *AgentSpeak*, originalmente criada por Anand Rao[Rao 1996]. Para a interação com o ambiente foi utilizado o *framework* CArtAgO [Ricci et al. 2006]. O CArtAgO possibilita desenvolver e executar ambientes baseados em artefatos, estruturados em espaços de trabalho abertos (possivelmente distribuídos em toda a rede) que os agentes de diferentes plataformas podem se unir para trabalhar juntos dentro desses ambientes.

Durante este relatório são apresentados os planos e objetivos detalhados que serão executados por alguns agentes, como também dois experimentos para demonstrar o funcionamento do sistema. E por fim, segue uma breve descrição sobre as lições aprendidas durante o desenvolvimento, dificuldades e observações gerais.

2. DESENVOLVIMENTO DO TRABALHO

Serão apresentados diferentes cenários, com configurações diversas para exemplificar alguns comportamentos, de acordo com as quantidades de artefatos e demais configurações. Os agentes do sistema são divididos entre 5 tipos, sendo que cada tipo possui suas próprias características. Os tipos são os seguintes:

- Capitão: são os agentes que comandam o barco e que iniciam a navegação pelo Oceano.
- Pescador: são os agentes responsáveis pela pescaria, jogando as redes e capturando os peixes do Oceano.
- AuxPesca: responsáveis por recolher e jogar a âncora do barco e por descarregar os peixes quando estiver no Porto.
- Motorista: são os agentes responsáveis por comandar o caminhão e iniciarem o carregamento de peixes no Porto.
- Carregador: responsáveis por carregar os peixes no Caminhão do Porto e descarregar do Caminhão no Distribuidor.

A Figura 1 apresenta uma abstração da simulação que será realizada neste trabalho.

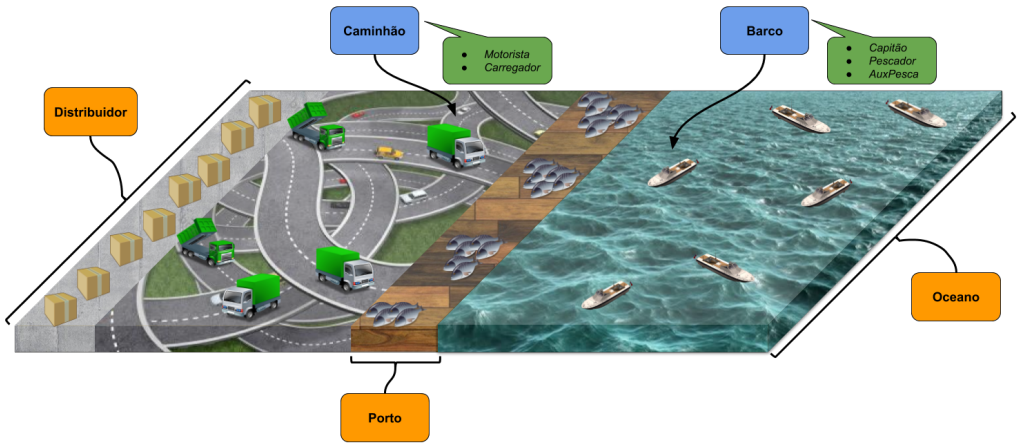

Figura 1. Visão abstrata do ambiente da simulação

Para elaborar um ambiente dentro desta abstração, foram criados artefatos para auxiliarem os agentes a interagirem neste ambiente e a conseguirem executar suas tarefas. A Figura 2 apresenta os artefatos deste ambiente e suas relações.

2.1. Desenvolvimento da Aplicação Jason

Para este relatório, serão abordados apenas os planos dos agentes Capitão e Motorista, pois eles são os responsáveis por iniciarem as simulações, cada um em seu lado, e acionar as ações dos demais agentes. Os planos dos agentes Pescador, AuxPesca e Carregador podem ser consultados em detalhe no código disponibilizado desta aplicação.

O sistema é iniciado logo que é verificado que existem peixes no Oceano. A partir disso, o Capitão inicia o plano de iniciar navegação, criando um artefato barco, verificando se há peixes no Oceano e se sua tripulação está completa, conforme apresenta a Figura 3.

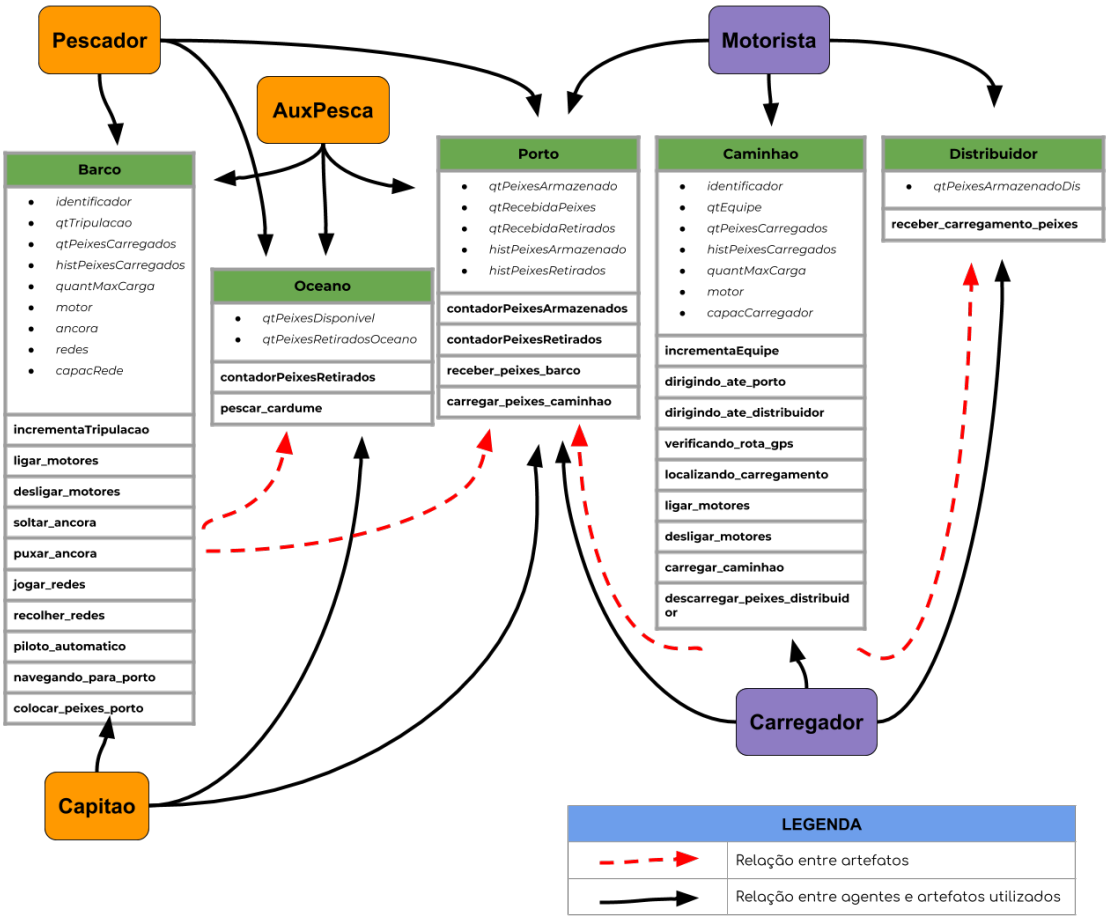

Figura 2. Artefatos utilizados pelos agentes no sistema

```
+!iniciarNavegacao
    : .my_name(NomeAgent) & identificador(IdBarco)
    <-
    .concat("barco_0", IdBarco, NomeBarco);
    .print("Nome do Barco - Capitao: ", NomeBarco);
    lookupArtifact(NomeBarco, ArtId);
    focus(ArtId);
    incrementaTripulacao(IdBarco, NomeAgent);
    .print("Eu sou o: ", NomeAgent, " do: ", NomeBarco);
    +meuBarcoEh(NomeBarco, IdBarco);
    !verificarPeixesOceano;
.
+!verificarPeixesOceano
    : qtPeixesDisponivel(QuantPeixesDisp) & QuantPeixesDisp > 0 & qtTripulacao(QuantTripbarco) & QuantTripbarco < 3
    <-
    !montarTripulacao;
.
+!verificarPeixesOceano
    : qtPeixesDisponivel(QuantPeixesDisp) & QuantPeixesDisp > 0 & qtTripulacao(QuantTripbarco) & QuantTripbarco == 3
    <-
    ?nomeAuxPesca(NomeAuxPesca);
    .print("CAPITÃO verifica com o AUX_PESCA se ele está pronto para iniciar a Pescaria...");
    .send(NomeAuxPesca, achieve, iniciar_pescaria);
.
```

Figura 3. Planos responsáveis por iniciar a navegação do barco

O plano da Figura 4, faz com que o Capitão seja responsável pela criação da sua tripulação, ou seja, cria um agente Pescador e um AuxPesca.

Após o artefato barco estar criado e a tripulação completa, o Capitão verifica com a sua tripulação (Pescador e AuxPesca) se eles estão prontos para iniciar a pescaria no Oceano. Para isso foram criados dois planos, que podem ser observados na Figura 5, onde o Capitão envia, em um deles, um *achieve* pedindo para o Pescador iniciar a pescaria, logo após já ter recebido a resposta do AuxPesca.

```
+!montarTripulacao
    : meuBarcoEh(NomeBarco, IdBarco)
    <-
    .concat("pescadorAg", IdBarco, NomePescador);
    .concat("auxPescaAg", IdBarco, NomeAuxPesca);
    .create_agent(NomePescador, "pescador.asl");
    .create_agent(NomeAuxPesca, "aux_pesca.asl");
    //Adiciona à sua base de crença o nome de seus companheiros de tripulacao
    +nomePescador(NomePescador);
    +nomeAuxPesca(NomeAuxPesca);
    .print("criou o agente: ", NomePescador);
    .print("criou o agente: ", NomeAuxPesca);
    .send(NomePescador, tell, meuBarcoEh(NomeBarco, IdBarco));
    .send(NomeAuxPesca, tell, meuBarcoEh(NomeBarco, IdBarco));
    !verificarPeixesOceano;
.
```

Figura 4. Plano responsável pela criação da tripulação do Barco

```
+!montarTripulacao
    : meuBarcoEh(NomeBarco, IdBarco)
    <-
    .concat("pescadorAg", IdBarco, NomePescador);
    .concat("auxPescaAg", IdBarco, NomeAuxPesca);
    .create_agent(NomePescador, "pescador.asl");
    .create_agent(NomeAuxPesca, "aux_pesca.asl");
    //Adiciona à sua base de crença o nome de seus companheiros de tripulacao
    +nomePescador(NomePescador);
    +nomeAuxPesca(NomeAuxPesca);
    .print("criou o agente: ", NomePescador);
    .print("criou o agente: ", NomeAuxPesca);
    .send(NomePescador, tell, meuBarcoEh(NomeBarco, IdBarco));
    .send(NomeAuxPesca, tell, meuBarcoEh(NomeBarco, IdBarco));
    !verificarPeixesOceano;
.
```

Figura 5. Planos responsáveis por verificar com os demais agentes se podem iniciar a pescaria

Com as respostas do Pescador e AuxPesca, o Capitão aciona o plano de ligar os motores do barco e inicia a busca por cardumes no Oceano. Os planos responsáveis por estas ações seguem na Figura 6. O plano localiza cardumes envia um pedido ao AuxPesca para que o agente jogue as redes no cardume.

```
+!ligar_motor <-
    .print("...ligando os motores para iniciar a navegacao em mar aberto...");
    ligar_motores; //função do artefato Barco
    !localizar_cardume;
.

+!localizar_cardume <-
    ?nomePescador(NomePescador);
    .print("...se preparando para localizar os cardumes no mar...");
    piloto_automatico;  //função do artefato Barco
    .print("cardume encontrado...");
    .print("CAPITÃO pede para o PESCADOR jogar as redes no Oceano...");
    .send(NomePescador, achieve, jogar_redes);
.
```

Figura 6. Planos responsáveis por ligar os motores e localizar os cardumes no Oceano

Quando o barco atinge a sua capacidade máxima de armazenamento de peixes, o AuxPesca avisa ao Capitão que precisam voltar ao Porto para descarregar. A Figura 7 mostra os planos de navegar para o Porto e o de desligar os motores. O plano de desligar os é responsável também por enviar um pedido ao AuxPesca para soltar a âncora do barco para descarregar.

```
+!nav_porto <-
    .print("...navegando em direção ao Porto para descarregar...");
    navegando_para_porto; //função do artefato Barco
    !desligar_motores;
.

+!desligar_motores <-
    ?nomeAuxPesca(NomeAuxPesca);
    .print("...desligando os motores...");
    desligar_motores; //função do artefato Barco
    .print("CAPITÃO pede para o AUX_PESCA soltar as ancoras no Oceano...");
    .send(NomeAuxPesca, achieve, soltar_ancora);
.
```

Figura 7. Planos responsáveis por navegar até o Porto e desligar os motores para descarregar

Enquanto isso, em terra, o agente Motorista verifica se existem peixes a serem carregados no Porto e também inicia a sua simulação, criando um artefato Caminhão conforme o plano da Figura 8.

```
+!iniciarTransporte
    : .my_name(NomeAgent) & identificador(IdCaminhao)
    <-
    .concat("caminhao_0", IdCaminhao, NomeCaminhao);
    .print("Nome do Caminhão - Motorista: ", NomeCaminhao);
    lookupArtifact(NomeCaminhao, ArtId);
    .print("ArtId do Caminhão - Motorista: ", ArtId);
    focus(ArtId);
    incrementaEquipe(IdCaminhao, NomeAgent);
    .print("Eu sou o: ", NomeAgent, " do: ", NomeCaminhao);
    +meuCaminhaoEh(NomeCaminhao, IdCaminhao);
    !verificarPeixesPorto;
.

+!verificarPeixesPorto
    : qtPeixesArmazenado(QuantPeixesArm) & QuantPeixesArm > 0 & qtEquipe(QuantEqCaminhao) & QuantEqCaminhao < 2
    <-
    !montarEquipe;
.

+!verificarPeixesPorto
    : qtPeixesArmazenado(QuantPeixesArm) & QuantPeixesArm > 0 & qtEquipe(QuantEqCaminhao) & QuantEqCaminhao == 2
    <-
    ?nomeCarregador(NomeCarregador);
    .print("MOTORISTA verifica com o CARREGADOR se ele está pronto para iniciar o Carregamento...");
    .send(NomeCarregador, achieve, buscar_peixes_porto);
.
```

Figura 8. Planos responsáveis por iniciar o transporte até o Porto

A seguir, o Motorista monta o seu time de agentes (Carregador), pelo plano de montar equipe, da Figura 9.

```
+!montarEquipe
    : meuCaminhaoEh(NomeCaminhao, IdCaminhao)
    <-
    .concat("carregadorAg", IdCaminhao, NomeCarregador);
    .create_agent(NomeCarregador, "carregador.asl");
    //Adiciona à sua base de crença o nome de seus companheiros de equipe
    +nomeCarregador(NomeCarregador);
    .print("criou o agente: ", NomeCarregador);
    .send(NomeCarregador, tell, meuCaminhaoEh(NomeCaminhao, IdCaminhao));
    !verificarPeixesPorto;
.
```

Figura 9. Plano responsável por montar a equipe do Caminhão

Quando o Motorista termina de criar seu time, ele solicita ao agente Carregador se preparar para iniciarem o deslocamento até o Porto para realizarem o carregamento de peixes. Com a resposta do Carregador positiva, o Motorista aciona o seu plano para se deslocar até o Porto, conforme observado na Figura 10.

Ao chegar no Porto, o Motorista procura por carregamento de peixes, aciona o

```
+!dirigir_ate_porto
    <-
    .print("...dirigindo ate o porto para buscar carregamento de peixes...")
    dirigindo_ate_porto; //função do artefato Caminhao
    !desligar_caminhao;
.

+!desligar_caminhao
    <-
    ?nomeCarregador(NomeCarregador);
    .print("...desligando o caminhao...")
    desligar_motores; //função do artefato Caminhao
    .print("MOTORISTA pede ao CARREGADOR que localize o carregamento de peixes no Porto...");
    .send(NomeCarregador, achieve, localizar_carregamento);
.
```

Figura 10. Planos responsáveis pelo deslocamento até o Porto e desligar os motores quando chegar

Carregador para carregar o Caminhão e voltar novamente ao Distribuidor descarregar. Os planos responsáveis por estas ações, seguem apresentados na Figura 11.

```
+!ligar_caminhao
    <-
    .print("...ligando o motor do caminhao para ir em direção ao Distribuidor...")
    ligar_motores; //função do artefato Caminhao
    !verificar_rota_gps;
.

+!verificar_rota_gps
    <-
    .print("...verificando o melhor caminho até o distribuidor...")
    verificando_rota_gps; //função do artefato Caminhao
    !dirigir_ate_distribuidor;
.

+!dirigir_ate_distribuidor
    <-
    .print("...dirigindo ate o distribuidor...")
    dirigindo_ate_distribuidor; //função do artefato Caminhao
    !descarr_peixes;
.

+!descarr_peixes <-
    .print("...retirando peixes do Caminhão e descarregando no Distribuidor...");
    lookupArtifact(distribuidor, ArtId); //busca o identificador do artefato Distribuidor
    descarregar_peixes_distribuidor(ArtId); //função do artefato Caminhão
    !verificando_peixes_porto;
.
```

Figura 11. Planos responsáveis por ligar os motores, dirigir até o Distribuidor e descarregar os peixes

Quando o Caminhão estiver parado vazio no Distribuidor, o Motorista faz uma requisição ao Porto para verificar se há peixes para serem carregados, caso não tenha, ele fica parado no Distribuidor. Essa requisição ocorre apenas uma vez, no exato momento que o Caminhão está no Distribuidor. Caso neste instante não tenha peixes, o Caminhão só vai voltar a carregar, quando receber um sinal do Porto avisando que algum carregamento de peixes foi inserido no Porto.

A simulação apresentada anteriormente executa de maneira interrupta. Para os experimentos desta aplicação, será utilizado um período de tempo pré-determinado, com configurações diferentes e em seguida serão verificados os resultados obtidos.

3. EXPERIMENTOS

Para demonstrar o resultado do desenvolvimento deste trabalho, foram realizados dois experimentos que apresentam simulações de uma configuração do funcionamento de uma peixaria, conforme apresentado na Tabela 1.

Tabela 1. Configurações das Simulações

	Artefatos	Ident.	Capac. Redes	Capac. Armazen.	Artefatos	Ident.	Capac. Carregador	Capac. Armazen.
Simulação 01	*Barco01*	1	25	200	*Caminhao01*	1	25	200
Simulação 02	*Barco01*	1	25	50	*Caminhao01*	1	25	100
	Barco02	2	25	50	*Caminhao02*	2	10	100

As simulações foram executadas por um período de 10 minutos cada, e para coletar as informações, foram criadas variáveis ”históricas”, com o intuito de registrar as informações de quantidade de peixes coletados e armazenados por cada artefato capaz de realizar estas ações.

3.1. Simulação 01

Nesta simulação foram criados um Barco e um Caminhão com capacidades de carregamento e de armazenamento iguais. Esta configuração quer verificar se a quantidade de peixes armazenados e coletados pelo Barco e pelo Caminhão são próximas quando suas configurações são idênticas.

Após a execução desta simulação, os resultados históricos foram mapeados na Tabela 2 para que seja possível realizar comparações.

Tabela 2. Resultados da Simulação 01

Artefatos	Histórico de Peixes Retirados	Histórico de Peixes Carregados	Histórico de Peixes Armazenados
Barco01	-	-	65400
Caminhao01	-	65200	-
Oceano	65400	-	-
Porto	65200	65400	-
Distribuidor	-	-	65200

Para esta configuração, os resultados mostram que quando os artefatos Barco e Caminhão possuem as mesmas configurações, as quantidades de de peixes retirados, carregados e armazenados, ficam muito próximas. A única observação desta primeira observação foi que quando a simulação foi parada, o Caminhão ainda estava em deslocamento ao Porto e por isso a quantidade de peixes carregados no Caminhão era menor que a quantidade de peixes carregados no Porto.

3.2. Simulação 02

Para esta configuração foram criados dois Barcos e dois Caminhões com capacidades de carregamento e de armazenamento diferentes. As capacidades dos Barcos foram configuradas de maneira igual, enquanto que a capacidade de carregamento dos Caminhões foi

configurada de maneira diferente. Esta estratégia foi utilizada com o objetivo de verificar se mesmo com Barcos com capacidades idênticas e capacidade de armazenamento inferior aos dos Caminhões, a quantidade de peixes coletada pelos Caminhões ainda é inferior ao dos Barcos.

Com a execução desta simulação, a Tabela 3 apresenta os resultados históricos mapeados para que seja possível realizar novas comparações.

Tabela 3. Resultados da Simulação 02

Artefatos	Histórico de Peixes Retirados	Histórico de Peixes Carregados	Histórico de Peixes Armazenados
Barco01	-	-	25950
Barco02	-	-	26000
Caminhao01	-	34615	-
Caminhao02	-	13865	-
Oceano	52000	-	-
Porto	51900	51950	-
Distribuidor	-	-	51380

Conforme pode ser observado nos resultados da segunda simulação, mantendo os Barcos com as mesmas capacidades, a quantidade de peixes armazenados por ambos, continua se mantendo muito próxima, mas como suas capacidades de armazenamento são menores, eles precisam se deslocar mais vezes entre Oceano e Porto para pescar e descarregar. Enquanto que com Caminhões com capacidades maiores de armazenamento, os deslocamentos entre Porto e Distribuidor são menores e por isso, suas quantidades de peixes carregados são bem maiores, diferente da primeira simulação.

Após a execução de ambas as simulações, conclui-se que a capacidade de armazenamento dos artefatos Barco e Caminhão é a variável independente predominante nos resultados. Mesmo que suas capacidades de redes ou carregamento sejam pequenas, o que importa nestas simulações é o armazenamento de cada um destes artefatos. Ou seja, o Barco pode jogar redes várias vezes, mas para ser eficiente, ele tem que ir menos vezes no Porto, da mesma forma que o Caminhão e o Distribuidor. Demais informações podem ser consultadas diretamente no sistema.

4. Considerações Finais

Esta aplicação foi desenvolvida durante a disciplina de Agentes Autônomos e demonstrou, de maneira prática, como utilizar os conceitos de programação orientada a agentes. O autor deste artigo já havia realizado a disciplina de Multiagentes e portanto, algumas das funcionalidades fornecidas pelo Jason e pelo CArtAgO já haviam sido exploradas anteriormente.

Durante o desenvolvimento desta aplicação foi utilizado como base o mesmo contexto criado na disciplina de Multiagentes (Peixaria). No entanto, sem o *framework* Moise [Hannoun et al. 2000], que é responsável pela organização dos agentes em sistemas multiagentes. Ao retirar o Moise da aplicação, todas as estratégias criadas anteriormente, planos, ações e crenças, tiveram que sofrer algum tipo de mudança, para que os agentes conseguissem se organizar e interagir entre si.

Na aplicação anterior, os agentes executavam suas interações uma única vez e em seguida paravam a simulação. Para esta aplicação, os agentes não param de realizar suas ações enquanto o sistema não é finalizado. Desta forma, o aplicação atual se torna mais completa, já que fica possível simular várias configurações, por períodos de tempo variados.

As dificuldades encontradas durante o desenvolvimento deste trabalho foram muito parecidas com o desenvolvimento do anterior, principalmente na programação da linguagem Jason.

O desenvolvimento também trouxe grandes aprendizados para o autor na área de programação orientada a agentes. Ao final deste desenvolvimento a sintaxe da linguagem Jason ficou muito mais clara para o autor, do que na disciplina anterior.

Como trabalho futuro, a interação no Oceano poderia ser repensada, para que os Barcos naveguem por mais tempo a procura de cardumes e também poderia ser criado um artefato ”Estrada”, para que os Caminhões interagissem mais durante o percurso de carga e descarga de peixes.

Referências

Alvares, L. O. and Sichman, J. S. (1997). Introduçao aos sistemas multiagentes. In *XVII Congresso da SBC-Anais JAI'97*.

Bordini, R. H., Hübner, J. F., and Wooldridge, M. (2007). *Programming multi-agent systems in AgentSpeak using Jason*, volume 8. John Wiley & Sons.

Demazeau, Y. and Müller, J.-P. (1990). *Decentralized Ai*, volume 2. Elsevier.

Hannoun, M., Boissier, O., Sichman, J. S., and Sayettat, C. (2000). Moise: An organizational model for multi-agent systems. In *Advances in Artificial Intelligence*, pages 156–165. Springer.

Rao, A. S. (1996). Agentspeak (l): Bdi agents speak out in a logical computable language. In *European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, pages 42–55. Springer.

Ricci, A., Viroli, M., and Omicini, A. (2006). Cartago: A framework for prototyping artifact-based environments in mas. In *International Workshop on Environments for Multi-Agent Systems*, pages 67–86. Springer.