ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

Avenida Professor Luciano Gualberto, travessa 3 nº 158 CEP 05508-900 São Paulo SP
Telefone: (011) 818-5583 Fax (011) 818-5294

Departamento de Engenharia de Computação e Sistemas Digitais

Aplicação de programação orientada à organização de agentes no Multi-Agent Programming Contest

Mariana Ramos Franco e Rafael Barbolo Lopes

PCS 5703 - Sistemas Multi-Agentes

Resumo Este trabalho apresenta o projeto e implementação de um SMA para participação no *Agent Contest* através da aplicação de programação orientada à organização de agentes. Para isso utilizou-se o modelo organizacional MOISE+ e a plataforma JASON.

1 Introdução

O objetivo do trabalho é projetar e implementar um Sistema Multi-Agente (SMA) para participar do *Agent Contest* de 2009.

O *Multi-Agent Programming Contest* é uma competição em que simultaneamente, dois times são situados no mesmo ambiente e competem diretamente num cenário definido pelos organizadores. Por se tratar de uma competição direta, é um cenário interessante para avaliar e comparar diferentes sistemas, assim possibilitando identificar pontos fortes e fracos de cada sistema, promovendo e avaliando o desenvolvimento de todos os participantes.

Conforme citado em [1], o *Multi-Agent Programming Contest* tem as seguintes características:

> *"This competition is an attempt to stimulate research in the area of multiagent systems development and programming by:*
>
> 1. *Identifying key problems,*
> 2. *Collecting suitable benchmarks, and*
> 3. *Gathering test cases which require and enforce coordinated action that can serve as milestones for testing multi-agent programming languages, platforms and tools."*

Em 2009, o *Agent Contest* definiu como tema para a competição um cenário de pastoreio, em que os agentes (comboys) estão localizados em um grid com árvores, vacas, cercas, currais e botões (que permitem a abertura da cerca). Árvores e currais são elementos estáticos, enquanto as cercas são controladas pelos agentes por meio de botões. Já as vacas possuem um algoritmo de movimentação com base nos elementos em torno delas, de modo a sofrerem atração mútua (tendendo a se agrupar) e a serem repelitas pelos obstáculos e agentes. A intensidade de cada um desses parâmetros comportamentais pode variar para cada cenário e é definida pelos organizadores do *Agent Contest*, não estando disponível diretamente para os competidores.

2 Análise e Especificação do SMA

Para o desenvolvimento do SMA foi utilizado um método situacional construido sob medida para o projeto, a partir de fragmentos de métodos capturados de três abordagens de desenvolvimento: USDP [3], Gaia [4] e MOISE+ [2].

O método é composto de quatro fases situacionais: Especificação de Requisitos, Análise, Design e Implementação.

Na fase de Especificação de Requisitos foram definidos os atores e os casos de uso do sistema. Já a fase de Análise foi composta de três subfases: Descrição do Ambiente em Gaia, Descrição da Organização em MOISE+, e Descrição das Interações usando Gaia.

2.1 Especificação de Requisitos

Identificação dos Atores

Ator	**Papel**	**Uso do sistema**
cowboy	Capturar vacas	Move-se pelo pasto, interage com vacas, abre/fecha cerca, interage com cowboy líder
cowboy sabotador	Atrapalhar equipe adversária	Move-se pelo pasto, interage com vacas, abre/fecha cerca da outra equipe, interage com cowboy líder
cowboy líder	Coordenar cowboys	Move-se pelo pasto, interage com vacas, abre/fecha cerca, coordena cowboys
vaca	Pastar	Move-se pelo pasto, se aproxima de rebanhos, se afasta de cowboys e de obstáculos

Observação: o ator vaca é implementado pelo simulador e por isso casos de uso iniciados por ele não serão apresentados neste documento.

Identificação dos Casos de Uso

Caso de uso	**Breve descrição**
Procurar vaca	Cowboys movem-se pelo pasto até encontrar uma vaca
Capturar vaca	Ao encontrar uma vaca, cowboys organizam-se para capturá-la
Atrapalhar equipe adversária	O cowboy sabotador atrapalha a equipe adversária
Abrir/fechar cerca	Cowboy se aproxima (ou se afasta) do botão para abrir (ou fechar) a cerca

Descrição dos Casos de Uso

Caso de Uso:	**Procurar vaca**
Ator:	cowboy, cowboy líder, vaca.
Início:	Cowboy líder deseja encontrar uma vaca.

Fluxo Típico	
No	**Ação**
1	Cowboys e comboy líder movimentam-se para uma nova posição.
2	Cowboy encontra uma vaca na vizinhança.
3	Cowboy grava o identificador e a coordenada da vaca encontrada.
Fluxos Alternativos	
Alternativa 1: Nenhum cowboy encontrou vaca no movimento.	
No	**Ação**
2	Cowboy não encontra uma vaca na vizinhança.
3	Repete passo 1.

Caso de Uso:	**Capturar vaca**
Ator:	cowboy, cowboy líder, vaca.
Início:	Cowboy encontrou uma vaca.

Fluxo Típico	
No	**Ação**
1	Cowboy emite o identificador e a coordenada da vaca para o cowboy líder.
2	Cowboy líder emite ordem de captura com identificador e coordenada da vaca para todos os cowboys.
3	Cowboys fazem formação um ao lado do outro para empurrar a vaca para a cerca (cowboy líder comanda o movimento).
4	Cowboys encurralam vaca na cerca.
5	Cowboy líder manda o cowboy mais próximo do botão abrir a cerca.
6	Vaca entra na cerca.
7	Cowboy fecha a cerca.
8	Cowboys desfazem formação.

Caso de Uso:	**Atrapalhar equipe adversária**
Ator:	cowboy sabotador.
Início:	Cowboy sabotador caminha até a cerca do adversário.

Fluxo Típico	
No	**Ação**
1	Cowboy sabotador chega na cerca do adversário.
2	Cowboy sabotador envia estímulos para o botão da cerca para mantê-la sempre aberta.

Diagrama do Modelo de Casos de Uso

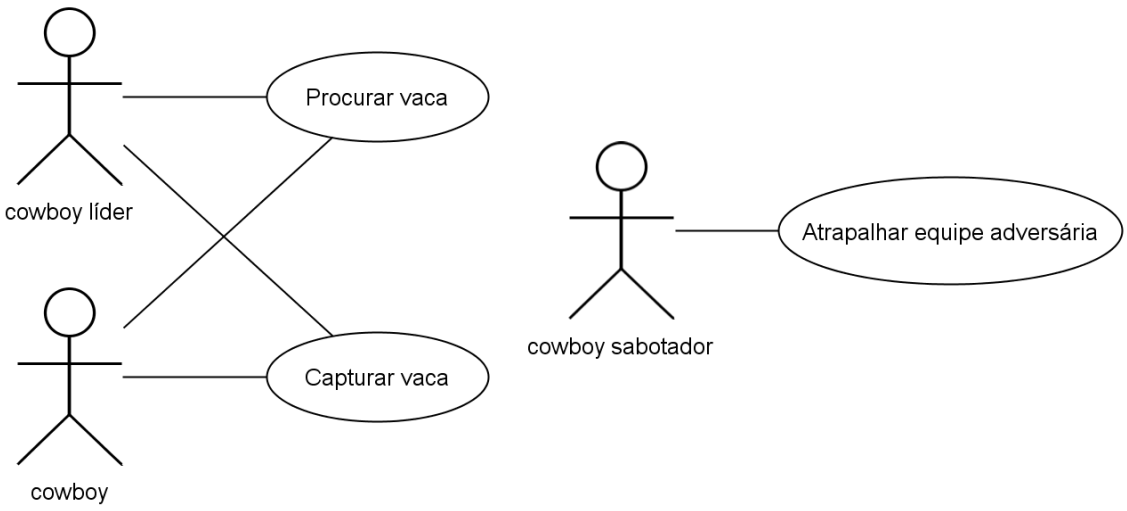

Figura 1. Diagrama do Modelo de Casos de Uso

2.2 Análise do SMA

Descrição do Ambiente (Gaia)

O ambiente em que o SMA atuará é composto de uma grade de células. O tamanho da grade é especificado no início da simulação e é variável. Porém, ele não pode ter mais de 150 x 150 células. A coordenada [0, 0] da grade está no canto superior esquerdo (noroeste). O ambiente simulado possui dois currais retangulares – um para cada equipe – que servem como um local para onde as vacas devem ser dirigidas. Além disso, há cercas que podem ser abertas através de botões.

Cada célula da grade pode ser ocupada por exatamente um dos seguintes objetos:

- Agente (controlado e capaz de se mover de uma célula para outra adjacente);
- Obstáculo (bloqueia a célula);
- Vaca (controlada pelo simulador, pode se mover de uma célula para outra adjacente, tende a formar rebanhos em áreas abertas e mantém distância de obstáculos e de agentes);
- Cerca (pode ser aberta usando um botão, para isto o agente deve ficar parado na célula adjacente ao respectivo botão).

Vacas devem ser empurradas para os currais. Cada time aprende a posição e as dimensões do seu curral no início de cada simulação.

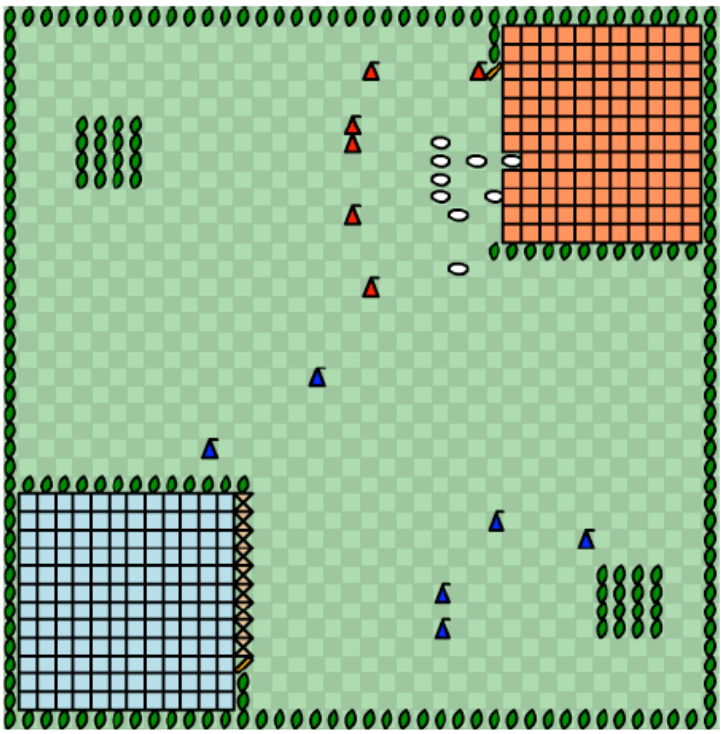

Figura 2. Ambiente de simulação. Agentes (triângulos vermelhos e azuis) são dirigidos pelos competidores. Obstáculos (círculos verde) bloqueiam células. Vacas (círculos brancos) movimentam-se de acordo com um algoritmo de vaca. Cercas (formas em x) podem ser abertas por um agente parado na célula adjacente ao botão (em forma de barra). As vacas têm que ser empurradas para dentro dos currais (retângulos vermelhos e azuis.)

Descrição da Organização (MOISE+)

O propósito da organização de agentes é capturar o maior número possível de vacas e atrapalhar o time adversário.

Existem 3 grupos de agentes:

- Liderança: grupo composto pelo cowboy líder que coordena os outros cowboys;
- Sabotagem: grupo composto pelo cowboy sabotador que procura atrapalhar a equipe adversária;
- Captura: grupo composto pelos cowboys responsáveis por capturar as vacas.

Existem 3 metas a serem cumpridas pelos agentes: procurar vaca, capturar vaca e atrapalhar equipe adversária. Para cada uma dessas metas, há missões específicas que os agentes devem cumprir. A especificação funcional no item 3.1 apresenta essas metas e missões.

Descrição das Interações (Gaia)

O modelo de interações representa as dependências e relacionamentos entre os papéis do sistema multiagentes, de acordo com as definições de um protocolo de comunicação entre os agentes.

As comunicações entre os agentes modelados ocorrem apenas entre os agentes com papel de capturador e o agente com papel de coordenador. Mais detalhes são dados no item 3.2.

3 Arquitetura e Design do SMA

A fase de design (projeto) do SMA especificada no método situacional é composta de quatro subfases: Modelo de Organização (MOISE+), Modelo de Interações (Gaia), Projeto de Agentes (Gaia) e Projeto de Comportamento Organizacional (MOISE+).

3.1 Modelo de Organização (MOISE+)

MOISE+ define 3 tipos de especificação para o modelo e projeto de organizações em um SMA:

- Especificação Estrutural: define os papéis, relações entre papéis e grupos.
- Especificação Funcional: define as missões e planos.
- Especificação Deôntica: especifica, no nível individual, como permissões e obrigações de um papél estão relacionas com uma missão.

Especificação Estrutural

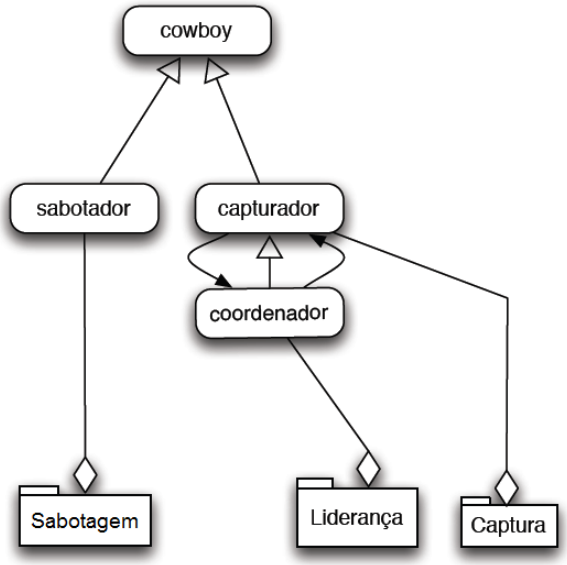

Figura 3. Especificação Estrutural

Papel de origem	Papel de destino	Tipo de link
Coordenador	Capturador	Autoridade
Capturador	Coordenador	Comunicação

Links entre papéis

Especificação Funcional

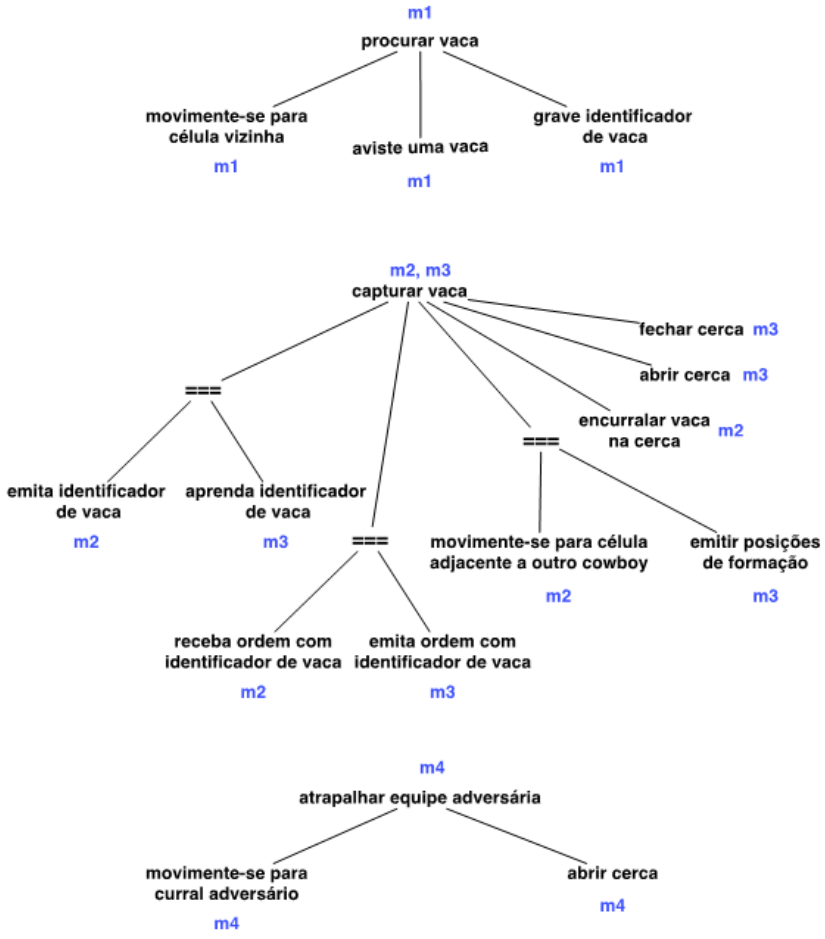

Figura 4. Especificação Estrutural

Especificação Deôntica

Papel	Relação deôntica	Missão	Restrição temporal
Sabotador	Obrigação	m4	A qualquer momento
Capturador	Obrigação	m1	Enquanto procura vaca
Capturador	Obrigação	m2	Enquanto captura vaca
Coordenador	Obrigação	m1	Enquanto captura vaca

3.2 Modelo de Interações (Gaia)

As comunicações entre os agentes modelados ocorrem apenas entre os agentes com papel de capturador e o agente com papel de coordenador.

A seguir definimos os possíveis protocolos de comunicação entre os agentes:

Protocolo: Encontrou vaca		
Iniciador: comboy	Parceiro: cowboy líder	Entrada: identificador e coordenada da vaca encontrada.
Descrição: Comboy informa o líder que encontrou uma vaca.		Saída OK, dados recebidos.

Protocolo: Cercar vaca		
Iniciador: cowboy líder	Parceiro: cowboy	Entrada: identificador e coordenada da vaca a ser capturada; coordenada para onde o cowboy deve se locomover para começar a cercar a vaca.
Descrição: Líder ordena o cowboy capturar vaca.		Saída OK, dados recebidos.

Protocolo: Abrir/Fechar cerca		
Iniciador: cowboy líder	Parceiro: cowboy	Entrada: vaca no curral.
Descrição: Líder ordena o cowboy a abrir/fechar a cerca.		Saída OK, cerca aberta/fechada.

3.3 Projeto de Agentes (Gaia)

O time de agentes para a competição será sempre formado de:

- 1 comboy líder
- 1 comboy sabotador
- 1 à N comboys

onde N é o número máximo de agentes no time - 2.

3.4 Projeto de Comportamento Organizacional (MOISE+)

No sistema multiagente proposto, não é possível que agentes entrem ou saiam do ambiente. Cada time é composto de um número fixo (definido pelo simulador) de cowboys que permanecem no ambiente durante toda a simulação.

Os agentes recebem os papéis no início da simulação. Um agente receberá o papel de coordenador, e liderará a procura e captura de vacas. Outro agente receberá o papel de sabotador, e terá como missão

atrapalhar o time adversário a atingir seus objetivos. Os outros agentes serão capturadores, e agirão em conjunto na procura e captura de vacas.

Todos os agentes do sistema estão compromissados com pelo menos uma meta global do sistema(capturar vacas e atrapalhar o adversário). As metas locais desses agentes foram projetadas para alcançar as metas globais. Logo, o comportamento dos agentes sempre será dirigido por missões locais que visam as metas globais.

4 Linguagens de programação e plataforma de execução

Para o desenvolvimento do SMA, foram utilizadas as linguagens XML, Java e AgentSpeak. Para a execução do SMA, foi utilizado a plataforma JASON e como ambiente de desenvolvimento, a IDE Eclipse com o plugin JASON instalado. A seguir, cada uma dessas tecnologias será apresentada com seu papel no projeto.

A linguagem de marcação XML foi utilizada para definir a especificação organizacional do SMA em MOISE+. Foram descritas as especificações estrutural, funcional e normativa do SMA em XML.

A linguagem Java foi utilizada para implementar o socket de comunicação com o servidor de simulação do *Multi-Agent Programming Contest* de 2009, assim como para implementar muitas das funcionalidades dos agentes, tais como: *InternalActions* - ferramentas escritas em Java para fazer cálculos que seriam ineficientes em AgentSpeak; *WorldModel* - modelo de mundo do agente; *CowboyArch* - arquitetura dos agentes. Por exemplo, a funcionalidade de movimento de um agente foi implementada em Java, através de cálculo de rota e troca de mensagens com o servidor de simulação.

A linguagem AgentSpeak foi utilizada para implementar os agentes. Para cada agente, foram definidas suas crenças, regras e metas iniciais, assim como as missões às quais ele estaria comprometido à realizar. Também foram implementados eventos organizacionais, estruturais e funcionais para cada agente, assim como seus planos.

O interpretador JASON foi utilizado para executar e depurar o SMA desenvolvido. Ele realiza a conexão entre a organização MOISE+ definida em XML, as implementações dos agentes em AgentSpeak e as implementações de suas funcionalidades em Java. Para facilitar o desenvolvimento, foi utilizada a IDE Eclipse com o plugin JASON instalado.

5 Estratégia para time de agentes

A estratégia do nosso time de agentes consiste em:

- Capturar pelo menos 1 vaca durante a partida;
- Impedir que o time adversário capture vacas.

Para garantir a captura de pelo menos 1 vaca, foi implementado um esquema em MOISE+ de captura de vacas que consiste em procurar uma vaca e guardá-la dentro do curral. Para impedir que o time adversário capture vacas, foi implementado um esquema que garante que a cerca do time adversário esteja sempre aberta, tornando-se impossível assim guardar a vaca dentro do curral.

Conforme as mensagens são recebidas do servidor, o agente atualiza o seu modelo de mundo (*WorldModel*) com o conteúdo de cada célula ao seu redor e, no caso dos cowboys, repassa essas informações para o cowboy líder. Dessa forma o líder passa a ter uma visão mais completa sobre o ambiente.

Inicialmente os agentes andam livremente sobre o mapa, procurando sempre ir para a célula ao seu redor menos visitada. O algoritmo de deslocamento dos agentes consiste em receber a posição atual do agente e a posição final desejada. Com isso, é possível obter o sentido de movimento que deve ser percorrido (*norte*, *nordeste*, *leste*, *sudeste*, *sul*, *sudoeste*, *oeste* ou *noroeste*). Antes de executar o movimento, é verificado se não há um obstáculo na posição que será alcançada. Quando há obstáculo, o agente se move aleatoriamente em outro sentido, sem que ocorra bloqueio de seu movimento.

A procura de uma vaca consiste em um agente cowboy se movimentar pelo ambiente até sentir a presença de uma vaca. Após encontrar a vaca, o agente informa ao agente líder a localização desta vaca. Ao notar a existência de um cluster de vacas o agente líder define a formação à ser tomada pelos cowboys e a captura das vacas é iniciada. A captura do cluster de vacas é feita em grupo pelos agentes cowboys. O cowboy líder envia para cada agente a posição à ser tomada e os cowboys encurralam e empurram o cluster de vacas em sentido ao curral. Enquanto isso o cowboy líder se movimenta até a cerca do curral para mantê-la aberta e assim as vacas poderem entrar.

Para impedir que o time adversário capture vacas, um cowboy sabotador se move até a cerca do time adversário e a mantém sempre aberta. Desta forma, é improvável que uma vaca capturada da equipe adversária fique presa dentro da cerca.

5.1 Inconsistências entre projeto e implementação

Durante o desenvolvimento do projeto, foram realizados testes com o ambiente de simulação que revelaram algumas falhas na especificação do projeto. Essas falhas geraram pequenas inconsistências entre a especificação do projeto e a implementação final, que serão apresentadas a seguir.

Uma falha identificada durante testes com o servidor de simulação foi a abertura e o fechamento das cercas. Não estava explicitado que a cerca ficava aberta apenas enquanto um cowboy estivesse ao lado de seu botão. Desta forma, para que um cowboy consiga passar pela cerca, é necessário que um outro cowboy fique ao lado do botão de abertura e fechamento da cerca. Havia sido suposto que o botão da cerca era um interruptor e não um botão de pressão. Assim, foi necessário modificar a implementação do esquema de atrapalhar a equipe adversária para que dois cowboys, ao invés de apenas um, caminhem até a cerca do inimigo e a mantenham sempre aberta.

Foi implementado apenas um esquema em MOISE+ para procura e captura das vacas, e não dois como especificado na fase de design. Assim, definimos a missão de encontrar uma vaca como a missão inicial no único esquema de captura. Além disso, para simplicidade da implentação, decidiu-se em diminuir o número de missões. O esquema adotado foi o seguinte:

```
<scheme id="catchCowScheme" >
   <goal id="catchCow" min="1">
      <plan operator="sequence">
         <goal id="search_cow" ds="search a cow" type="maintenance"/>
         <goal id="enclose_cows" ds="herd_cows">
            <plan operator="parallel">
               <goal id="coordinate_cowboys" type="maintenance"/>
               <goal id="herding_cows" type="maintenance"/>
            </plan>
         </goal>
      </plan>
   </goal>

   <mission id="mission1" min="1"> <!-- cowboys mission -->
      <goal id="search_cow" />
      <goal id="herding_cows" />
   </mission>
   <mission id="mission2" min="1" max="1"> <!-- leader mission -->
      <goal id="coordinate_cowboys" />
   </mission>
</scheme>
```

6 Características técnicas

O sistema desenvolvido apresentou instabilidade de comunicação com o servidor de simulação, pois este aguarda um *timeout* (tempo de resposta) grande antes de executar as ações dos agentes em um único passo.

Desta forma, alguns agentes enviam uma sequência de mensagens que nem sempre são recebidas pelo servidor de simulação, o que gera ciclos de processamento em inanição.

Testamos a situação em que a simulação é interrompida e o nosso sistema reinicia a partida do ponto que parou. Alguns problemas foram observados neste caso. Cada agente do SMA possui uma representação do mundo, que é construída no decorrer da simulação. Ao interromper a simulação, todo o conhecimento de mundo acumulado é destruído (pois este está armazenado em memória dinâmica). Ao reiniciar a simulação em um ponto diferente, é comum que os agentes levem um certo tempo até aprenderem novamente as informações do mundo e darem continuidade à execução de seus objetivos de forma coordenada.

Foi observado que o sistema consome uma baixa quantidade de memória, porém uma elevada taxa de processamento. A baixa quantidade memória ocorre porque as estruturas de dados armazenam informações muito simples, como pequenas cadeias de caracteres e inteiros. A elevada taxa de processamento ocorre porque o servidor de comunicação não é em tempo real, o que faz com que muitos ciclos com trocas de mensagem sejam cancelados. Além disso, temos várias threads rodando ao mesmo tempo para processar as mensagens trocados com o sevidor, atualizar o modelo de mundo dos agentes e calcular a próxima ação à ser executada.

7 Discussão e conclusão

O método situacional adotado mostrou-se muito completo nas fases de análise, especificação e design. Através das atividades propostas, foi possível especificar muito bem o SMA e construir corretamente os modelos propostos na fase de design. Na fase de implementação, o método situacional forneceu uma sequência lógica de atividades que contribuiu para a implementação dos agentes e do ambiente. Exceto pela fase de implementação, todas as outras foram seguidas, atividade por atividade, durante o desenvolvimento do SMA.

O modelo organizacional baseado em MOISE+ foi muito bem aplicado ao domínio do SMA. Não houve dificuldade em mapear as estruturas e funcionalidades do SMA para o modelo organizacional adotado. A implementação da organização também foi fácil utilizando JASON.

Através das simulações de execução do nosso time de agentes, verificamos que a sabotagem da equipe adversária funciona bem tecnicamente. É necessário verificar se esta estratégia funcionará bem em uma competição real. Para melhorar a sabotagem da equipe adversária, seria interessante criar mais agentes que atrapalham o processo de captura de vacas da equipe adversária (por exemplo, agentes que empurram as vacas no sentido contrário à entrada da cerca).

Houve dificuldade em encontrar recursos online para aprendizado de programação de SMA usando JASON. Os artigos online eram muito simples e a documentação da API de JASON para Java era complexa e incompleta. Para a disseminação de tecnologias multiagentes, seria interessante o desenvolvimento de recursos online de aprendizado mais acessíveis, principalmente para plataformas mais populares, como o JASON.

Com o desenvolvimento do projeto para o *Multi-Agent Programming Contest*, ganhamos experiência com a plataforma JASON, que, apesar de complexa, se mostrou muito poderosa e flexível para a programação de SMAs. O projeto contribuiu muito com o entendimento de como são realizadas interações entre os diversos agentes de um SMA, ficando claro que não é possível controlar sequências de ações de um grupo de agentes ou mesmo de um único agente. Também foi muito útil para obter experiência prática com organização de agentes utilizando MOISE+.

Referências

1. Multi Agent Programming Contest Home. `http://www.http://multiagentcontest.org/`.
2. Jomi Hubner, Jaime Sichman, and Olivier Boissier. A model for the structural, functional, and deontic specification of organizations in multiagent systems. 2507:439–448, 2002.
3. Ivar Jacobson, Grady Booch, and James Rumbaugh. *The unified software development process*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.
4. Franco Zambonelli, Nicholas R. Jennings, and Michael Wooldridge. Developing multiagent systems: The gaia methodology. *ACM Trans. Softw. Eng. Methodol.*, 12:317–370, July 2003.

A Projeto fornecido vs Projeto final

Foi fornecido aos grupos um projeto inicial com o ambiente, arquitetura dos agentes, comunicação com o servidor e algumas *InternalActions*. Deveríamos assim, nos limitarmos apenas em implentar a parte em AgentSpeak e a organização em Moise+.

Infelizmente, acabamos recebendo esse projeto inical com atraso. Como já tínhamos desenvolvido muita coisa, decidimos manter o que havia sido feito:

- Comunicação com o servidor: classes *ServerConnection* e *Messages*.
- Ambiente: classe *CowboysEnv*.
- Arquitetura dos agentes: classe *CowboyArch*.

E aproveitamos do projeto inicial as seguintes classes:

- Modelo de mundo dos agentes: classes *LocalWorldModel* e *WorldModel*.
- Algumas *InternalActions*: pacote *jia*.
- Classes de cluster de vacas: pacote *env.cow*.

Alguns ajustes precisaram ser feitos na arquitetura dos agentes e no ambiente para que essas classes do projeto inicial fossem aproveitadas. Infelizmente, não foi possível utilizar o componente resposável pela visualização gráfica dos agentes.