TITAN: Um jogo de estratégia simulado utilizando conceitos de sistemas multiagentes

Maicon R. Zatelli[1], **José R. F. Neri**[1], **Daniela M. Uez**[1], **Rafael F. Callegaro**[1]

[1] Departamento de Automação e Sistemas
Universidade Federal de Santa Catarina (UFSC) – Florianópolis, SC – Brasil

{maicon,dmuez}@das.ufsc.br, {jrf.neri, rafaelfrizzocallegaro}@gmail.com

Resumo. *O paradigma de agentes está sendo cada vez mais utilizado na área de desenvolvimento de jogos. As características dos agentes, como autonomia, proatividade e interação social, podem auxiliar no desenvolvimento de jogos mais realistas e detalhados. Este trabalho tem como objetivo integrar diversas ferramentas e conceitos da área de IA para o desenvolvimento de um jogo de estratégia. Além de agentes, foram utilizados os conceitos de organização, ambiente, reputação, sistemas especialistas e ontologias.*

1. Introdução

O paradigma de agentes tem sido cada vez mais utilizado pelos desenvolvedores de jogos. As características dos agentes, como autonomia, proatividade e interação social, podem auxiliar no desenvolvimento de jogos mais realistas e detalhados [Carvalho 2004]. Além disso, jogos são cenários perfeitos para permitir o uso conjunto de diversos conceitos de Inteligência Artificial (IA). Esta é a principal motivação que levou ao desenvolvimento deste trabalho: integrar diversas ferramentas e conceitos da área de IA para solução de um problema. Desta forma optou-se pelo desenvolvimento de um jogo de estratégia, chamado Titan. O desenvolvimento do jogo ocorreu no contexto de uma disciplina e utilizou-se duas diferentes arquiteturas de agentes (reativa e BDI), os agentes foram desenvolvidos em diferentes linguagens de programação (JADE e Jason) e também foram utilizados os conceitos de organização, ambiente, reputação, ontologias e sistemas especialistas.

Este documento está estruturado em seis seções. Na primeira foi dada uma breve introdução a respeito do objetivo deste artigo. Na segunda é apresentada uma visão geral sobre o jogo. Na terceira é descrito como o jogo foi desenvolvido, detalhes da implementação e onde cada ferramenta foi utilizada. Na quarta são mostrados os principais desafios enfrentados durante o desenvolvimento. Por fim, na quinta seção, são enfatizadas as principais contribuições e na última seção estão as considerações finais.

2. Definição do Problema

Titan é um jogo de estratégia onde as equipes disputam o domínio de Titan, um satélite que orbita o planeta Saturno. A história se passa no futuro, quando os recursos na Terra escassearam e os humanos buscam em outros lugares do universo os recursos que faltam. O principal objetivo do jogo é garantir o domínio sobre Titan através da destruição das equipes inimigas. Titan possui diversos poços de recursos que as equipes devem encontrar e explorar. É pela obtenção destes recursos que é possível ampliar os exércitos. Cada equipe é composta por um ou mais times aliados, que trocam informações sobre o ambiente e sobre os adversários. Por sua vez, um time é composto por uma base, um conselho,

capitães, soldados, robôs, naves e exploradores. Além disso, cada time possui um *blackboard*, que nada mais é que um repositório de informações globais usado com o objetivo de permitir que os agentes compartilhem informações com os demais agentes do time, e um sistema especialista (SE), para auxiliar nos planos de ataque.

Inicialmente, um time tem uma base, que é responsável pelo gerenciamento dos seus recursos e também pela criação do conselho. O conselho é um agente que tem como responsabilidade organizar a defesa e o ataque, obedecendo às estratégias definidas pelo capitão e negociadas com os aliados. No decorrer do jogo, guerreiros são criados de acordo com as necessidades de ataque ou defesa de cada time, se a base possuir quantidade suficiente de recursos para criá-los. Os guerreiros sabem quem são seus aliados, movem-se pelo ambiente, atiram, reconhecem os inimigos, os poços e os obstáculos existentes. Existem cinco tipos de guerreiros: robô, nave, explorador, soldado e capitão. Cada guerreiro tem uma função diferente no cenário do jogo e também características distintas, conforme listado abaixo:

- Soldado: é o guerreiro responsável pela defesa da base e pelo cumprimento das missões de ataque;
- Robô: é um guerreiro que possui as mesmas responsabilidades do soldado, porém tem mais força do que ele;
- Nave: é um tipo de guerreiro que tem a capacidade de sobrevoar o ambiente. As naves têm um campo de visão maior do que os guerreiros terrestres e não são impedidas pelos obstáculos, podendo voar sobre os inimigos, bases e poços. Seu principal objetivo é explorar o ambiente, localizando as bases inimigas e poços;
- Explorador: é o guerreiro cuja função principal é extrair recursos dos poços encontrados no ambiente;
- Capitão: é o guerreiro que tem como responsabilidade planejar e negociar ataques juntamente com outros capitães aliados. O capitão tem acesso a uma base de conhecimento que permite a ele planejar os ataques e também pode, quando necessário, criar outros guerreiros.

Os guerreiros utilizam o *blackboard* para armazenar as informações sobre a localização dos poços, dos inimigos e dos aliados que são vistos no ambiente. Sempre que encontrar um poço ou um inimigo que ainda não tinha sido descoberto, os guerreiros disponibilizam essa informação no *blackboard* para o restante do time. Essas informações são usadas pelo capitão para, juntamente com o SE, decidir se um ataque deverá ou não ser realizado.

Diferentemente do que acontece com outros jogos desenvolvidos com agentes, Titan é um jogo simulado. Os times iniciam ataques, organizam defesas e buscam poços sem necessidade de intervenção de um jogador humano. A única interação humana no jogo acontece na configuração inicial do ambiente, onde o jogador define a quantidade de times e de equipes existentes e, se desejar, o SE utilizado pelos times. Depois de iniciado o jogo, as batalhas se desenrolam sem necessidade de intervenção humana até que uma das equipes seja vencedora.

3. Desenvolvimento

Os agentes foram implementados utilizando diferentes arquiteturas. A base é um agente reativo implementado em JADE [Telecom 2011], enquanto que os guerreiros são agentes

BDI implementados em Jason [Bordini et al. 2007]. Outros conceitos da área de agentes foram utilizados no desenvolvimento do jogo. O capitão, por exemplo, tem acesso a um SE para decidir quando os ataques devem ser realizados e um *blackboard* é utilizado para auxiliar os times a montar uma visão global do ambiente. Os agentes de times aliados cooperam entre si para realizar ataques contra as bases inimigas, os quais são negociados pelos capitães de cada time. Então o capitão repassa as decisões da negociação ao conselho, que é responsável por organizar as unidades de ataque para que realizem o ataque. Ao mesmo tempo, o conselho cria unidades de defesa a fim de garantir a defesa da própria base. Assim, ao mesmo tempo que existe uma hierarquia entre os agentes, muitas decisões também podem ser tomadas pelos guerreiros independentemente de ordens recebidas de um agente superior.

3.1. O ambiente

De forma geral, o ambiente é o espaço onde todos os agentes vivem, ou seja, onde percebem e atuam [Demazeau 1995]. O ambiente do jogo foi desenvolvido utilizando o CArtAgO [Ricci 2011]. Todo o ambiente é implementado dentro de um único artefato e nele estão os times, os agentes, os poços, os obstáculos, o *blackboard* e o SE. É neste artefato que está descrito o comportamento do ambiente, permitindo que os agentes realizem uma série de operações e percepções. Entre elas pode-se citar a locomoção pelo ambiente, extração de recursos, percepção de outros agentes ou bases, ataque a agentes ou bases, percepção de obstáculos, entre outros.

3.2. A organização

Numa organização há uma espécie de união entre os agentes que formam um grupo. A organização pode ser usada pra representar objetivos globais e os agentes passam a se relacionar para atingirem esses objetivos. Como metas locais e globais nem sempre são compatíveis, uma organização fornece uma visão geral para verificar se as ações dos agentes respeitam os objetivos globais ou não [Gers 2002].

Os agentes que formam um time precisam trabalhar conjuntamente para que o objetivo de destruir o inimigo seja alcançado. Para garantir a organização desse sistema utilizou-se o modelo organizacional Moise [Hubner 2003]. Conforme pode ser visto na Figura 1, na especificação estrutural foram especificados quatro grupos. O grupo `time` (grupo raiz) é composto pelos subgrupos `exploração`, `defesa` e `ataque`. Uma instância do grupo `time` deve possuir uma única instância do grupo `exploração`, uma do grupo `defesa` e uma do grupo `ataque`. Além disso, o grupo precisa ter um conselho mas não precisa necessariamente ter um capitão, já que, durante o jogo, o capitão pode ser morto e o grupo ficará sem capitão até que um novo seja criado. Os agentes do grupo `exploração` podem assumir o papel de nave ou o papel de explorador e a nave tem autoridade sobre o explorador. O grupo `defesa` é composto de zero ou infinitas unidades de defesa e o grupo `ataque` é composto de zero ou infinitas unidades de ataque.

Os papéis de explorador, nave, unidade de defesa, unidade de ataque e capitão herdam as características do papel abstrato guerreiro e todos os guerreiros podem se comunicar entre si. Agentes do tipo robô e soldado podem assumir o papel de unidade de ataque ou unidade de defesa, de acordo com a necessidade do time. O papel assumido por um agente é definido pelo conselho no momento em que o agente é criado. Se uma

unidade é criada e a defesa já está completa, a unidade assume o papel de ataque. Finalmente, o conselho tem autoridade sobre os guerreiros e estes têm conhecimento do conselho, enquanto que o capitão tem autoridade sobre as unidades de ataque e de defesa.

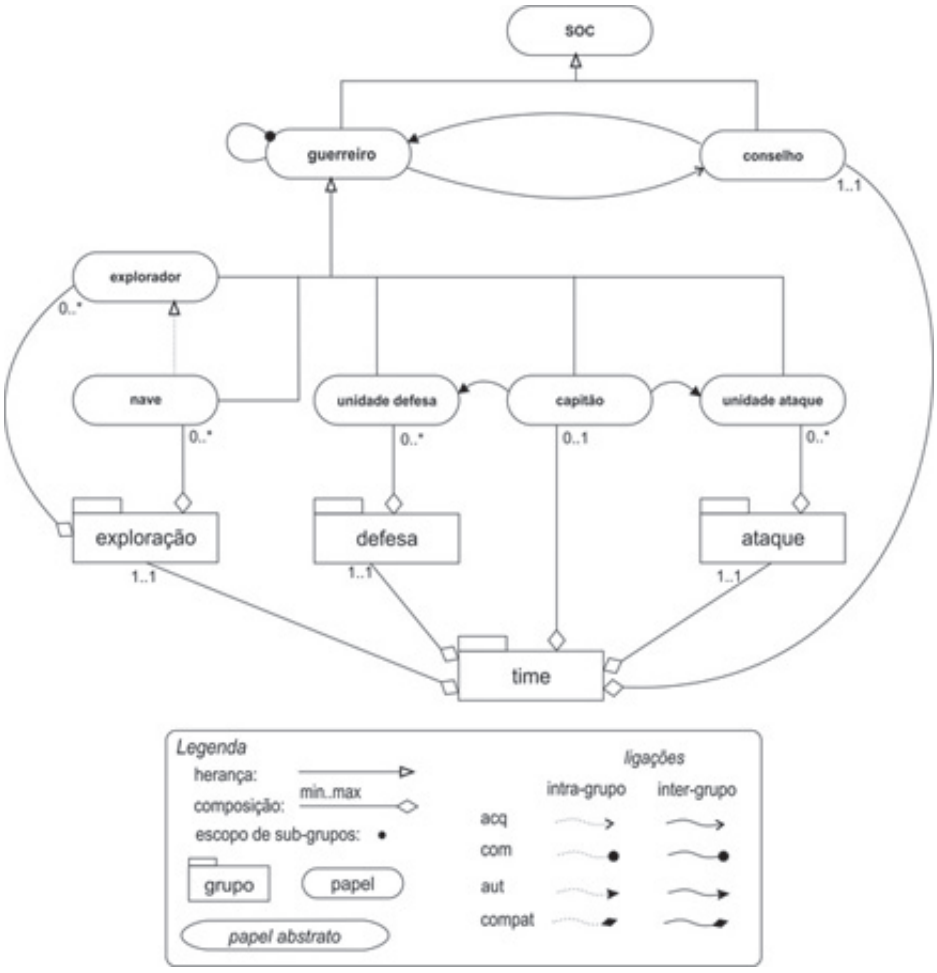
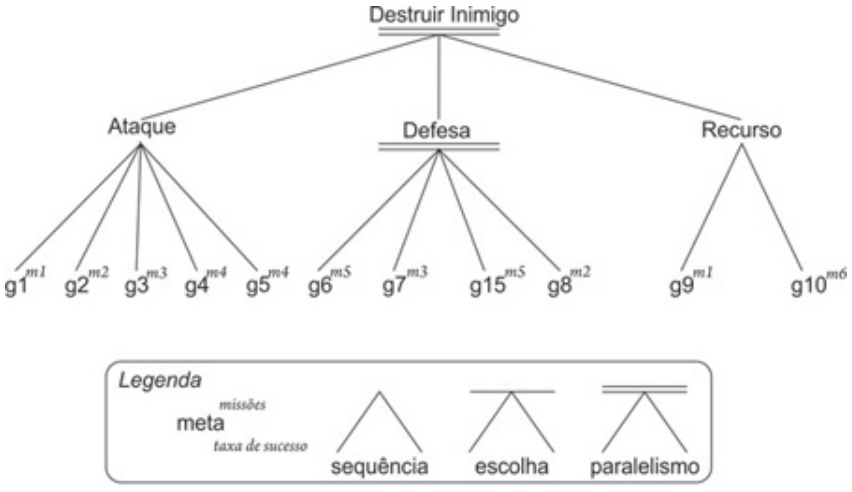


Figura 1. Especificação estrutural para o jogo `Titan`.

A Figura 2 apresenta um esquema social definindo a especificação funcional do jogo. Na raiz do esquema tem-se a meta global de destruir o inimigo. Para satisfazer a meta global, três submetas em paralelo precisam ser satisfeitas: a meta `ataque`, a meta `defesa` e a meta `recurso`. Essas três metas são decompostas em três planos, e esses planos são compostos por metas folhas. As metas dos planos `ataque` e `recurso` devem ser alcançadas em forma sequencial, ou seja, só é possível alcançar a meta `g2` depois que a meta `g1` for alcançada. Já as metas do plano `defesa` podem ser alcançadas de forma paralela, ou seja, pode-se alcançar a meta `g8` e `g6` ao mesmo tempo.

As missões são assumidas pelos agentes no momento em que estes assumem um papel em um grupo. Conforme especificado na Figura 2, o agente que se compromete com a missão `m1` é responsável pela satisfação de todas as metas desta missão, no caso, as metas `g1` e `g9`. Já o agente que assumir a missão `m2` deve alcançar as metas `g2` e `g8`.

A relação entre a especificação estrutural e especificação funcional é feita pela especificação normativa. Na especificação normativa são descritas as missões com as



meta	descrição
g1	encontrar base inimiga
g2	negociar ataque
g3	juntar nº X de soldados ataque
g4	ir até a base inimiga
g5	atacar inimigo ataque
g6	ficar próximo à base
g7	juntar nº X de soldados defesa
g15	atacar inimigo defesa
g8	negociar defesa
g9	encontrar recurso
g10	explorar recurso

Figura 2. Especificação funcional para o jogo.

quais um papel tem permissão ou obrigação de se comprometer. Na Figura 3 é possível observar que o papel de nave está obrigado a alcançar as metas da missão `m1`, o capitão tem obrigação de alcançar a missão `m2`, e assim por diante.

papel	relação deôntica	missão
nave	obrigação	m1
capitao	obrigação	m2
conselho	obrigação	m3
unidade ataque	obrigação	m4
unidade defesa	obrigação	m5
explorador	obrigação	m6

Figura 3. Especificação normativa para o jogo.

3.3. Comunicação entre agentes

Um dos aspectos mais importantes no desenvolvimento de um SMA é a interação entre os agentes. O principal mecanismo que permite esta interação é a comunicação através da troca de mensagens. Como sistemas multiagentes nem sempre são sistemas fechados, desenvolvidos por um único grupo de desenvolvedores, foi preciso criar protocolos padronizados objetivando que os agentes possam se comunicar com outros agentes mesmo que não tenham sido desenvolvidos especificamente para interagirem entre si. Algumas linguagens, como KQML, KIF e a FIPA-ACL, foram criadas com este fim. Neste traba-

lho utilizou-se duas dessas linguagens. A KQML é utilizada pela ferramenta Jason e a o padrão FIPA-ACL é utilizado pela ferramenta JADE.

A *Knowledge Query and Manipulation Language* (KQML) é uma linguagem externa para a comunicação de agentes. Essa linguagem define um formato de envelope para as mensagens, no qual um agente pode explicitar a força das intenções ilocucionárias da mensagem. Cada mensagem tem uma performativa, que pode ser qualquer classe de mensagens, e um número de parâmetros, compostos por pares de atributos e valores [Wooldridge 2009].

Por sua vez, a *Foundation for Intelligent Physical Agents*(FIPA) desenvolveu a linguagem FIPA-ACL que é similar à linguagem KQML. A linguagem FIPA-ACL também define uma linguagem externa para as mensagens, porém diferentemente da KQML, a FIPA ACL possui somente 20 performativas para a interpretação das mensagens e não obriga uma linguagem especifica para o conteúdo das mensagens [Wooldridge 2009].

Outra forma utilizada para comunicação entre os agentes é através do uso de um sistema de *blackboard*. Um sistema de *blackboard* é composto por uma coleção de entidades conhecidas como fontes de conhecimento, cada uma possuindo um conhecimento especializado, e uma estrutura de dados compartilhada que essas fontes de conhecimento utilizam para a comunicação. As fontes de conhecimento, neste caso os agentes, são capazes de ler e escrever no *blackboard*, assim contribuindo com uma solução parcial do problema [Wooldridge 2009].

No jogo, os agentes utilizam estas duas formas de comunicação: a troca de mensagens e o *blackboard*. A troca de mensagens é utilizada pelos agentes para comunicar o conselho quando nascem ou morrem, quando a base inimiga foi destruída e para incluir informações no *blackboard*. Além disso, os capitães usam mensagens para negociar os ataques com os aliados, os quais podem ser realizados com base nas informações do *blackboard* e com ajuda do SE.

3.3.1. Negociação

Negociação é uma técnica utilizada para que agentes possam chegar a um acordo sobre algum assunto de interesse mútuo. A negociação normalmente procede em uma série de etapas, com todos os agentes envolvidos fazendo uma proposta em cada etapa. As propostas dos agentes são feitas de acordo com a estratégia definida por eles, porém deve seguir um protocolo, ou seja, deve respeitar um conjunto de propostas possíveis em cada etapa. Se um acordo é alcançado então a negociação termina. Um exemplo de uma questão simples que envolve um cenário de negociação é onde dois agentes estariam negociando um preço de alguma coisa [Wooldridge 2009].

A fim de coordenar os diversos times aliados durante os ataques, o jogo utiliza uma espécie de negociação. A negociação é feita entre os capitães de times aliados. Um capitão só pode entrar em uma negociação se não estiver em outra negociação e se o time possuir unidades de ataque livres, ou seja, que não foram designadas para nenhuma missão. Antes de iniciar a negociação, o capitão precisa saber a quantidade de guerreiros conhecidos dos inimigos e quem são os comandantes aliados. De posse dessas informações o capitão inicia a negociação, que ocorre em quatro fases:

1. Início: o capitão informa aos aliados qual o inimigo que será atacado;
2. Inventário: levantamento da quantidade de guerreiros de todos os aliados que podem fazer parte do ataque;
3. Definição: consulta ao plano de ataque para saber se o ataque deve ser realizado ou não;
4. Ataque: envio de notificação às unidades para realizar o ataque.

A negociação é abortada em cada fase se algum aliado recusar a participar do ataque. Nesse caso, a negociação é reiniciada com os capitães que estão dispostos a colaborar. Por fim, sempre que a negociação for cancelada e reiniciada, todos os capitães participantes serão notificados.

3.3.2. Reputação dos agentes

Bromley define reputação como sendo uma ferramenta social que objetiva reduzir a incerteza na interação entre indivíduos com atributos desconhecidos [Bromley 1993]. A interação entre os agentes é um aspecto inerente aos sistemas multiagentes, visto que os agentes precisam interagir para que os objetivos globais ou particulares possam ser atingidos. Assim, é comum que os sistemas multiagentes utilizem modelos de reputação para auxiliar os agentes na hora de determinar quem será seu parceiro de negócios. Nesse sentido, a reputação pode ser interpretada como um valor, determinado pela sociedade, que demonstra o grau de confiabilidade de um determinado agente.

No jogo, a reputação foi utilizada para permitir que os capitães escolham, dentre seus aliados, aqueles que estão mais dispostos a cooperar com o ataque. A reputação é calculada da seguinte maneira: sempre que um time se dispuser a ajudar os aliados durante uma negociação, um ponto será acrescido à sua reputação. Se o time não puder ajudar, um ponto será descontado de sua reputação. Caso a negociação seja finalizada com sucesso, ou seja, se ao final da negociação os aliados decidirem atacar o inimigo, o time que iniciou o processo é recompensado com o aumento de um ponto de reputação para cada time aliado que participou do processo. Já no caso da negociação fracassar e não haver ataque, nenhum time participante é pontuado. O uso do sistema de reputação diminui a quantidade de mensagens trocadas e torna a negociação mais rápida. Nesse processo, times que não estão dispostos a colaborar têm uma reputação baixa e, portanto, não são contados pelos aliados no início da negociação.

3.4. Sistema Especialista

Um sistema especialista é um sistema que possui a capacidade de emular a habilidade de decisão de um especialista humano sobre um certo domínio de conhecimento. Um exemplo clássico de um SE é o MYCIN [Buchanan and Shortliffe 1984], que pretendia dar assistência no tratamento de infecções sanguíneas em humanos [Jackson 1998].

No jogo, o capitão tem acesso a um SE que o auxilia a definir se o ataque a determinada base inimiga deve ou não ser realizado, levando em conta a quantidade de guerreiros disponíveis, a quantidade de guerreiros que o inimigo possui, recursos, entre outros dados relevantes sobre os times. Cada time pode utilizar um SE próprio, o que diferencia o comportamento dos capitães durante a negociação. Para descrever o SE utilizou-se a ferramenta Jess [Friedman-Hill 2011].

3.5. Ontologia

Uma ontologia descreve os conceitos de um domínio de conhecimento e define relações entre eles. O principal objetivo da ontologia é capturar o conhecimento consensual sobre algum domínio de interesse. Studer define ontologia como ”uma especificação explícita e formal de uma conceitualização compartilhada”. Ela deve ser compreensível, acessível e manipulável pelos agentes que farão uso da mesma [Studer et al. 1998].

No jogo, a ontologia foi utilizada pelo agente base para que este soubesse qual tipo de papel cada um dos outros agentes pode assumir. Por exemplo, o papel de capitão só pode ser assumido pelo agente do tipo capitão, o papel de explorador só pode ser assumido pelo agente do tipo explorador ou nave. Essa restrição não pode ser feita a partir do Moise e é importante para o jogo pois um agente do tipo capitão não possui as características necessárias para assumir o papel de explorador.

Para o desenvolvimento da ontologia foi utilizada a ferramenta Protégé [Research 2011], e a integração com o sistema foi feita através do uso do *reasoner* Hemit [Group 2011], que tem a finalidade de efetuar o raciocínio da ontologia. Por meio do mecanismo de inferência disponibilizado pelo Hemit foi possível inferir quais tipos de agentes podem assumir cada papel. Para que seja possível trabalhar com este recurso foi elaborada uma ontologia que detalha os equipamentos e títulos que um agente deve possuir para que possa assumir determinado papel. A Figura 4 exibe a classe `Equipamento`, que especifica a lista de equipamentos disponíveis no jogo. Cada agente pode possuir desde zero ou até vários destes equipamentos. Já a Figura 5 permite visualizar a classe `Titulo`, que contém a lista de títulos que uma unidade pode obter. O título é usado para permitir que um agente possa assumir os papéis de `Capitão` ou `Conselho`, que não implicam na necessidade de um agente possuir algum equipamento específico.

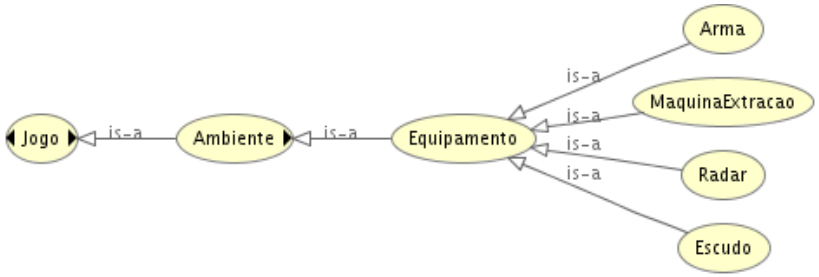


Figura 4. Classe `Equipamento`.

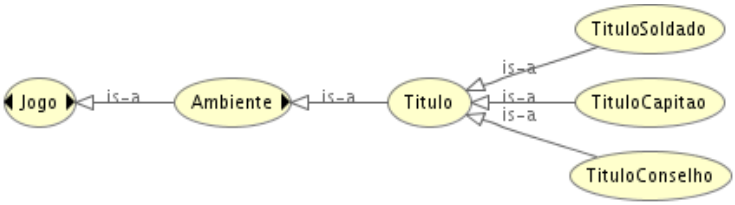


Figura 5. Classe `Título`.

Após a elaboração destas classes foi necessário relacionar elas por meio de propriedades. Como exemplo, um papel de ataque só pode ser assumido por um agente que

possua uma arma de equipamento, um papel de exploração só pode ser assumido por um agente que possua um radar ou uma máquina de extração de recursos, um papel de capitão só pode ser assumido caso o agente possua um título de capitão, e assim por diante. O diagrama de todas as inferências obtidas pelo *reasoner* Hemit entre papéis e agentes é mostrado na Figura 6.

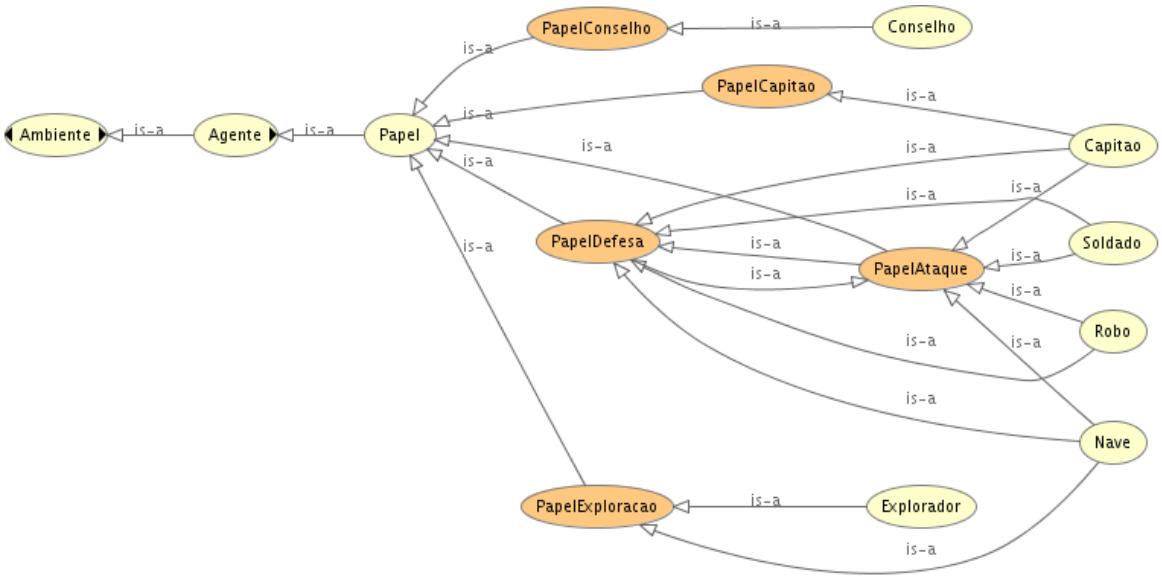


Figura 6. Ontologia inferida da classe `Agente`.

Na ontologia também foram implementadas a descrição dos artefatos contidos na classe `Artefato` e a relação de acesso a estes artefatos, ou seja, a definição de quais agentes possuem acesso um determinado artefato. A classe `Artefato` tem como objetivo fazer uma separação funcional do ambiente desenvolvido no CArtAgO, ilustrando que tipos de artefatos existem no jogo, visto que na implementação foi optado por utilizar um único artefato que controla todo o ambiente do jogo.

3.6. Interface Gráfica

Com a finalidade de visualizar o comportamento do jogo optou-se por desenvolver uma interface gráfica simples. Nesta interface, cada time é representado por uma cor. A Figura 7 mostra a descrição dos times que estão no jogo. Neste caso são quatro times: time 0, de cor azul, time 1, de cor verde, time 2, de cor vermelha e time 3, de cor cinza. Para cada time é apresentado a quantidade total de unidades, a distribuição dessas unidades por papéis e a lista de aliados, se houver. O time 0 (azul), por exemplo, não possui aliados. Já o time 1 (verde) possui dois aliados: o time 2 (vermelho) e o time 3 (cinza).

Na Figura 8 pode-se visualizar o ataque coordenado dos times verde e cinza contra a base do time azul. Nesta representação gráfica, os soldados e robôs de ambos os times aliados atiram contra a base. Os tiros são representados por linhas amarelas. As letras `r` representam robôs, os `s` representam soldados, o `c` representa o capitão, o círculo maior representa a base e o asterisco circundado representa um poço. Poços ainda não explorados são representados em amarelo, poços já explorados são representados na cor preta.

```
Id: 0 Unidades: 1 Mana: 200 Aliados: null
PapelConselho(0)\PapelDefesa(1)\PapelExploracao(0)\PapelCapitao(1)\PapelAtaque(1)
Id: 1 Unidades: 19 Mana: 14300 Aliados: 2 / 3
PapelConselho(0)\PapelDefesa(18)\PapelExploracao(1)\PapelCapitao(1)\PapelAtaque(18)
Id: 2 Unidades: 11 Mana: 460 Aliados: 1 / 3
PapelConselho(0)\PapelDefesa(11)\PapelExploracao(0)\PapelCapitao(1)\PapelAtaque(11)
Id: 3 Unidades: 20 Mana: 17420 Aliados: 1 / 2
PapelConselho(0)\PapelDefesa(19)\PapelExploracao(2)\PapelCapitao(1)\PapelAtaque(19)
```

Figura 7. Descrição dos times.

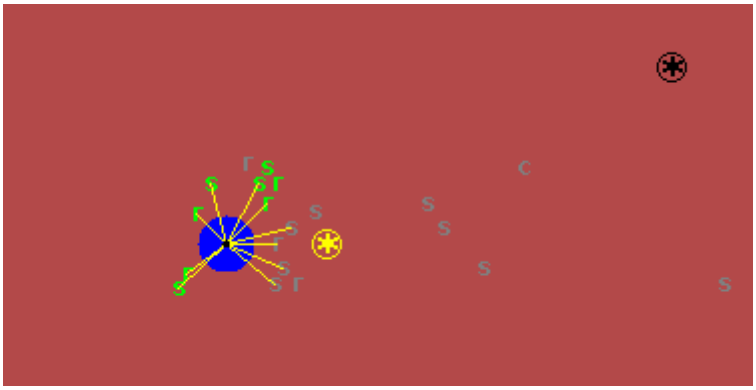


Figura 8. Visualização do jogo.

Cada agente tem seu campo de visão. O campo de visão determina a área na qual os objetos ou outros agentes podem ser vistos pelo agente. O tamanho do campo de visão depende do tipo de agente.

4. Desafios durante o desenvolvimento

Durante o desenvolvimento do jogo alguns problemas foram enfrentados. Por exemplo, quando diversos agentes precisavam acessar o ambiente desenvolvido na ferramenta CArtAgO e os acessos eram constantes, o ambiente tornava-se lento e acabava por não executar o jogo. Para corrigir o problema foi necessária uma alteração na ferramenta. No caso da interface gráfica desenvolvida em Java, a lentidão foi corrigida com a criação de uma *thread* especifica para atualizar da interface. Também visando um melhor desempenho do jogo optou-se por desenvolver um único artefato contendo todas as operações que os agentes podem executar no ambiente, ao invés de diversos artefatos.

A ferramenta JADE também apresentou alguns problemas. Em primeiro lugar, descobriu-se que o JADE não interage com o ambiente CArtAgO. Esse problema teve de ser contornado com a criação de uma representação do agente JADE no ambiente do CArtAgO. Essa foi a maneira encontrada para que um agente desenvolvido em JADE pudesse interagir com o ambiente. Mesmo assim, o agente JADE tem acesso somente a algumas percepções e ações. Essa restrição precisou ser colocada pois este agente não tem mecanismos de controle de concorrência. Nos demais agentes o acesso concorrente é controlado pelo CArtAgO. O agente JADE foi concebido como um agente reativo, já que a linguagem não oferece funcionalidades para criação de agentes BDI. A principal função

deste agente é gerenciar o *blackboard*, ou seja, para atualizar ou consultar o *blackboard* é necessário enviar mensagens para o agente JADE. Este procedimento visa evitar que uma quantidade grande de acessos ao ambiente seja feita, evitando que este se tornasse um gargalo para o sistema. Outro problema apresentado foi com relação a quando uma quantidade grande de mensagens era trocada. Neste caso, a execução se tornava lenta. Por isso acabou-se por reduzir o número de mensagens trocadas entre os agentes.

Por fim, o Moise apresentou problemas quando o agente era removido do sistema. Como o agente tem obrigação de realizar uma missão, ele não pode abandoná-la. Mesmo quando um agente é removido do sistema através da ação interna `kill_agent`, o Moise não libera o papel para outro agente. Para solucionar esse problema, foi feita uma alteração no esquema do Moise para permitir que os papéis tivessem um número indefinido de agentes. O Moise também apresentou problemas com missões do tipo *maintenance*, que são missões que sempre precisam ser executadas continuamente, e missões que precisam ser executadas diversas vezes pelo mesmo agente. Esses tipos de missão não são disponibilizadas pelo Moise.

5. Contribuições do trabalho

Devido aos problemas citados acima, foi possível realizar algumas colaborações para as ferramentas utilizadas. No projeto CArtAgO foi corrigido e reportado o problema que havia em relação a lista de eventos pendentes dos agentes, na qual estes eventos nunca eram apagados, causando certa instabilidade no jogo. Algumas contribuições foram dadas também ao projeto Moise, como a sugestão de desenvolvimento de metas do tipo *maintenance*. Esse tipo de meta deve ser executada pelo agente repetidas vezes até que este abandone seu papel. É uma meta que, na prática, nunca entra em um estado de satisfeita, mantendo-se sempre ativa. Outras colaborações para o projeto Moise foram reportar os problemas que ocorriam quando um agente abandonava o sistema, considerando principalmente o fato deste agente estar comprometido com uma meta obrigatória. Quanto aos problemas encontrados durante o uso da ferramenta JADE, descobriu-se que os desenvolvedores já têm conhecimento dessas falhas.

6. Considerações finais

Com o desenvolvimento do jogo foi possível observar o uso de cada conceito e ferramenta na prática. Como cada time pode utilizar um sistema especialista próprio, diferente dos demais, o sistema acabou por permitir times com comportamentos estratégicos distintos, tornando o jogo mais interessante. O uso do ambiente CArtAgO mostrou-se de grande ajuda pois, através da simulação de um ambiente virtual por meio de artefatos, foi permitido separar de maneira coerente o desenvolvimento dos agentes dos demais objetos envolvidos no sistema, os quais não possuem autonomia. Já o Jason demonstrou ser uma linguagem adequada para o desenvolvimento de agentes, o qual, diferentemente do JADE, permitiu a criação de agentes BDI de maneira bastante simples e intuitiva. Na dimensão da organização pôde ser aproveitada a ferramenta Moise, que apesar de alguns recursos ainda não estarem disponíveis, mostrou ser útil para controlar um grupo de agentes, levando-os a cumprir as metas de uma forma mais organizada e civilizada.

A ontologia permitiu o uso de um *reasoner* para mostrar ao usuário quais são os papéis disponíveis pela organização, porém também informar quantas unidades poderiam

assumir cada papel em tempo de execução. Dessa forma foi permitido ao usuário ter uma noção do que se passa durante o jogo em termos de números de unidades. Por fim, a reputação teve um papel importante durante a negociação, reduzindo significativamente a troca de mensagens entre capitães pertencentes a times que não estavam colaborando nos combates.

7. Agradecimentos

Agradecemos aos professores Jomi Fred Hübner (UFSC) e Ricardo José Rabelo (UFSC) por comentários e sugestões a respeito do trabalho.

Referências

Bordini, R. H., Wooldridge, M., and Hübner, J. F. (2007). *Programming Multi-Agent Systems in AgentSpeak using Jason (Wiley Series in Agent Technology)*. John Wiley & Sons.

Bromley, D. B. (1993). *Reputation, Image and Impression Management*. John Wiley & Sons.

Buchanan, B. G. and Shortliffe, E. H. (1984). *Rule Based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Carvalho, F. (2004). Comportamento em grupo de personagens do tipo black&white.

Demazeau, Y. (1995). From interactions to collective behaviour in agent-based systems. In *In: Proceedings of the 1st. European Conference on Cognitive Science. Saint-Malo*, pages 117–132.

Friedman-Hill, E. (2011). Jess, the rule engine for the java platform. Disponível em `http://www.jessrules.com/`.

Gers, F. (2002). *Multi-agent system for distributed data fusion in peer-to-peer environment.* PhD thesis, University of Jyväskylä.

Group, I. S. (2011). Hermit owl reasoner. Disponível em `http://hermit-reasoner.com/`.

Hubner, J. F. (2003). *Um Modelo de Reorganização de Sistemas Multiagentes*. PhD thesis, Escola Politécnica da Universidade de São Paulo.

Jackson, P. (1998). *Introduction to Expert Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3rd edition.

Research, S. C. F. B. I. (2011). Protégé. Disponível em `http://protege.stanford.edu/`.

Ricci, A. (2011). Cartago. Disponível em `http://cartago.sourceforge.net/`.

Studer, R., Benjamins, V. R., and Fensel, D. (1998). Knowledge engineering: Principles and methods. *Data and Knowledge Engineering*, 25(1-2):161–197.

Telecom, I. (2011). Jade - java agent development framework. Disponível em `http://jade.tilab.com/`.

Wooldridge, M. (2009). *An Introduction to MultiAgent Systems*. Wiley, 2nd edition.