Using the JaCaMo framework to develop a SMA for the MAPC 2012 "Agents on Mars" scenario

Mariana Ramos Franco, Jaime Simão Sichman
Laboratório de Técnicas Inteligentes (LTI)
Escola Politécnica (EP)
Universidade de São Paulo (USP)
Email: mafranko@usp.br, jaime.sichman@poli.usp.br

***Abstract*—This paper describes the architecture and core ideas of the Multi-Agent System created by the LTI-USP team which participated in the 2012 edition of the Multi-Agent Programming Contest (MAPC 2012). The contest scenario represented a coordinated exploration on Mars. The team organization was based on the *Moise* organisational model, and its implementation used the *JaCaMo* multi-agent framework.**

***Keywords*—Multi-Agent System, Multi-Agent Programming, JaCaMo, Jason, CArtAgO, Moise.**

I. Introdução

Recentemente, tem havido um movimento rumo ao uso de organizações explícitas no projeto e desenvolvimento de Sistemas Multiagentes (SMA) [1], [2]. A organização ajuda a modelar melhor o problema em questão, e a aumentar a eficiência do sistema, definindo a estrutura do SMA e as regras que os agentes devem seguir para atingir seus objetivos.

Seguindo esta premissa, neste artigo descrevemos um SMA com organização explícita, desenvolvido para participação na edição de 2012 do *Multi-Agent Programming Contest*[1] (MAPC 2012).

O MAPC é uma competição realizada todos os anos com o propósito de estimular a pesquisa no campo da programação de SMA [3]. Em cada rodada do evento, dois times de agentes 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, identificar pontos fortes e fracos, e promover o desenvolvimento de todos os participantes.

Nosso time foi baseado no modelo organizacional *Moise*, e foi desenvolvido utilizando a infraestrutura fornecida pelo arcabouço *JaCaMo*.

Moise[2][4] é um modelo organizacional que decompõe a especificação da organização em três dimensões: estrutural, funcional e deôntica. O modelo permite que o projetista especifique explicitamente as restrições e os padrões de cooperação a serem impostas ao agentes pela organização.

JaCaMo[3] [5] é um arcabouço que cobre todos os níveis de abstração necessários para o desenvolvimento de SMA sofisticados, pois possibilita, além da programação dos agentes, a definição do ambiente e da organização do SMA. Cada um desses níveis de abstração (agente, ambiente e organização) são implementados através da combinação de três tecnologias distintas: *Jason*[4] [6], para programação dos agentes; *CArtAgO*[5] [7], para programação do ambiente; e *Moise*, para programação da organização.

A motivação principal para a participação na competição foi testar e analisar o arcabouço *JaCaMo* e sua camada organizacional, a fim de identificar os pontos fracos e fortes da plataforma, e suas possíveis limitações de desempenho.

A seguir, o cenário da competição é brevemente descrito na seção II, e em seguida o SMA desenvolvido é apresentado na seção III. Alguns resultados preliminares de análise de desempenho do arcabouço *JaCaMo* são mostrados na seção IV e as conclusões do trabalho apresentadas na seção V.

II. *Multi-Agent Programming Contest*

O MAPC é uma competição internacional realizada anualmente desde 2005 com o propósito de estimular a pesquisa na área de programação de SMA. Em 2011, o MAPC definiu como tema para a competição o cenário "*Agents on Mars*", no qual os concorrentes devem projetar uma equipe de 20 agentes para explorar e ocupar as melhores zonas de Marte.

Neste cenário, dois times de agentes competem para dominar os melhores poços de água do planeta. O ambiente é representado por um grafo ponderado, em que os vértices denotam poços e possíveis localizações para os agentes; e as arestas indicam a possibilidade de atravessar de um vértice para outro, com um custo em energia para o agente. Cada vértice possui um valor correspondente ao poço de água nele localizado, e esse valor é utilizado para o cálculo do valor das zonas ocupadas pelos agentes.

Uma zona é um sub-grafo coberto por uma equipe de acordo com um algoritmo de coloração baseado na noção de domínio [3]. Vários agentes podem estar em um único vértice, mas o seu domínio é dado ao time com maior número de agentes. Se um conjunto de agentes domina um vértice por número, este recebe a cor da equipe dominante. Um vértice sem cor que tem a maioria dos vizinhos com uma cor específica herda essa cor para si. Por fim, se o grafo geral contém um sub-grafo colorido que constitui uma fronteira, todos os nós que estão dentro desta fronteira também são coloridos. Isto

[1]Site da competição: http://multiagentcontest.org/.
[2]Disponível em http://moise.sourceforge.net/.
[3]Disponível em http://jacamo.sourceforge.net/.
[4]Disponível em http://jason.sourceforge.net/.
[5]Disponível em http://cartago.sourceforge.net/.

significa que os agentes podem colorir ou cobrir um sub-grafo que tem mais vértices do que o número de agentes. A Figura 1 mostra parte de um mapa com os sub-grafos coloridos.

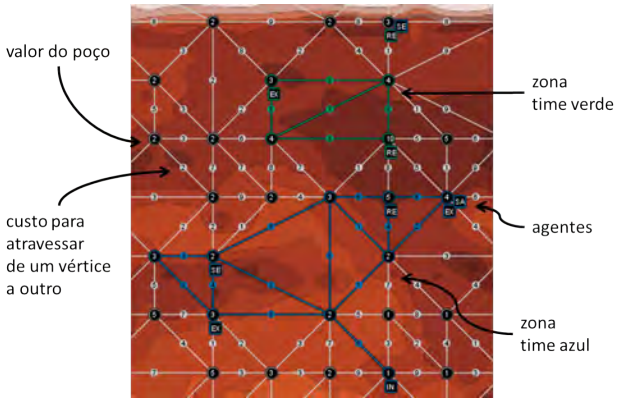

Figura 1. Cenário *"Agents on Mars"*.

O mapa é desconhecido dos agentes no começo da simulação. Assim, cada equipe precisa explorar o grafo antes de começar a conquistar as zonas, dado ainda que os agentes possuem visão limitada do mapa e só recebem percepções dos vértices próximos. Além disso, às vezes, uma equipe precisa sabotar o outro time para conseguir aumentar sua área, ou se defender para não perder zonas para o oponente.

Cada time é formado por 20 agentes e cinco papéis diferentes, sendo quatro agentes por papel. Os papéis, descritos na Tabela I, definem características próprias de cada agente, tais como nível de vida, energia máxima, força e visibilidade, e as ações que o agente pode realizar no ambiente. Por exemplo, os exploradores podem encontrar poços de água e ajudar a explorar o mapa, os sentinelas possuem sensores de longa distância e assim podem observar grandes áreas, os sabotadores podem atacar e desativar os inimigos, os inspetores podem espionar os adversários, e os reparadores podem consertar agentes danificados.

Se uma equipe atinge um marco importante, ela recebe uma recompensa em dinheiro. A recompensa ganha por uma equipe pode ser usada para equipar os agentes, aumentando, por exemplo, o máximo de energia ou a força de um agente. Existem diferentes marcos que podem ser atingidos durante uma competição, tais como ter zonas com valores fixos (por exemplo, 10 ou 20), número fixo de ataques bem sucedidos, ou número fixo de defesas realizadas com sucesso. Se não for usado, o dinheiro ganho é somado à pontuação total da equipe.

O objetivo do jogo é maximizar a pontuação do time, que é definida como a soma dos pontos obtidos pelas zonas ocupadas com o dinheiro ganho (e não gasto) em cada passo da simulação, como mostra a Equação 1:

$$pontuacao = \sum_{p=1}^{passos} (zonas_p + dinheiro_p) \qquad (1)$$

Tabela I. PAPÉIS E AÇÕES.

	explorador	reparador	sabotador	sentinela	inspetor
recarregar	x	x	x	x	x
atacar			x		
defender		x	x	x	
mover	x	x	x	x	x
sondar[6]	x				
examinar[7]	x	x	x	x	x
inspecionar[8]					x
comprar	x	x	x	x	x
reparar		x			

III. *LTI-USP Team*

A seguir, são apresentadas a arquitetura e as estratégias principais do SMA, chamado de "*LTI-USP Team*", que foi desenvolvido utilizando o arcabouço *JaCaMo* para participar do MAPC 2012.

O *JaCaMo* [5] é um arcabouço que cobre todos os níveis de abstração necessários para o desenvolvimento de sofisticados SMA, pois possibilita, além da programação dos agentes, a definição do ambiente e da organização do SMA. Cada um desses níveis de abstração (agente, ambiente e organização) são implementados através de um dos três componentes que compõem o arcabouço: *Jason*, *CArtAgO* e *Moise*.

A. *Arquitetura*

A arquitetura do SMA é apresentada na Figura 2 e baseia-se no modelo BDI [8]. Os agentes foram desenvolvidos através da plataforma *Jason*, e cada agente possui sua própria thread de controle, planos, uma base de crenças e modelo de mundo. Os planos são especificados em *AgentSpeak*[9], e em cada passo da simulação o agente decide qual plano será executado de acordo com as suas crenças e visão local do mundo.

O modelo de mundo consiste de um grafo desenvolvido em Java utilizando classes e estruturas de dados simples. Ele capta todos os detalhes recebidos do servidor do MAPC, tais como vértices e arestas exploradas, posição dos adversários, companheiros de equipe desativados, etc. Em cada passo da simulação, o modelo de mundo do agente é atualizado com as percepções recebidas do servidor e com as informações recebidas dos outros agentes. O agente pode acessar ou alterar o estado do seu modelo de mundo através de ações internas desenvolvidas em Java. Alguns exemplos de ações internas são:

- $closer\_repairer(Pos)$, que devolve ao agente a posição do agente reparador mais próximo,
- $move\_to\_target(Pos, Target, NextPos)$, que diz ao agente para qual vértice ele deve se mover a fim de alcançar determinada posição no grafo.

Algumas percepções recebidas do servidor são também armazenadas na base de crenças do agente, tais como seu

[6] A priori, os agentes não tem conhecimento sobre o valor dos poços de água. Apenas após sondar um vértice é que o time toma conhecimento do valor do poço.

[7] Inicialmente, os agentes não conhecem qual o custo de se atravessar de um vértice a outro. Para descobrir qual o valor de cada aresta, o agente precisa examiná-la antes.

[8] Esta ação coleta informações sobre os oponentes presentes em vértices vizinhos, tais como nível de vida, energia e papel.

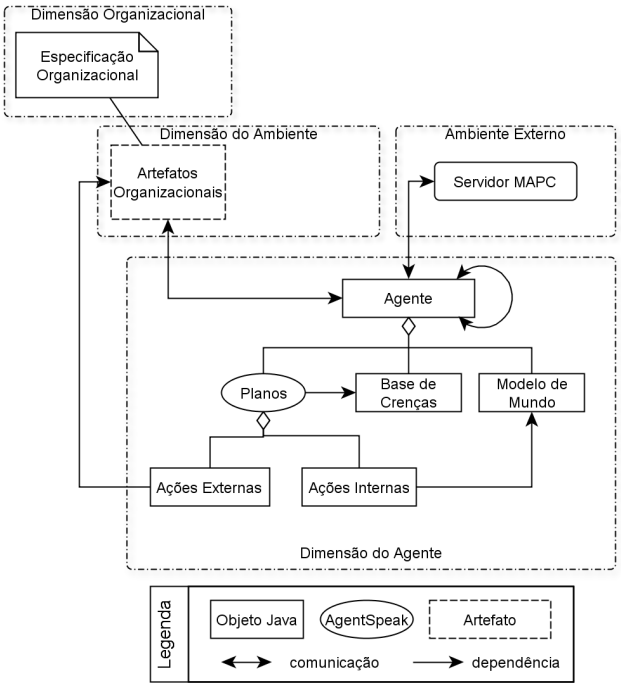

Figura 2. Arquitetura do *"LTI-USP Team"*.

papel, seu nível de energia e sua posição. Isto permite que o agente tenha acesso direto a essas informações sem ter que acessar o modelo de mundo. As percepções sobre vértices, arestas e outros agentes não são armazenadas na base de crenças para não comprometer o desempenho do agente, uma vez que poderia ser muito caro atualizar e acessar a base de crenças com tanta informação: o mapa de Marte utilizado na competição continha cerca de 400 vértices e centenas de arestas.

A comunicação entre os agentes ocorre através do *Jason* e, para reduzir a sobrecarga de comunicação, os agentes transmitem aos outros apenas as percepções novas, isto é, apenas as percepções recebidas do servidor que produzem alguma atualização no seu modelo de mundo. Por esta razão, existe uma forte troca de informação entre os agentes no início de uma partida, devido à comunicação sobre percepções novas, especialmente aquelas relacionadas à estrutura do mapa, como vértices e arestas. No entanto, a sobrecarga de comunicação diminui à medida que os agentes vão construindo um modelo de mundo mais completo. Os agentes se comunicam para:

(i) informar os outros agentes sobre a estrutura do mapa;

(ii) informar sobre agentes inimigos;

(iii) pedir que algum outro agente vá repará-lo;

(iv) pedir que um agente vá para determinado vértice.

Os agentes se comunicam com o servidor da competição através da interface EISMASSim incluída no pacote de software do simulador distribuído aos participantes. O EISMASSim se baseia no EIS[9][10], que é um padrão proposto para a interação agente-ambiente. Ele automaticamente estabelece e mantém conexões autenticadas com o servidor, além de abstrair a comunicação para simples chamadas de métodos em Java. A fim de utilizar essa interface, foi necessário estender a arquitetura padrão de agentes do *JaCaMo* para que fosse possível a um agente não só perceber e agir sobre os artefatos do *CArtAgO*, mas também interagir com o servidor.

CArtAgO é um arcabouço baseado no meta-modelo A&A (Agentes e Artefatos) [11] para o desenvolvimento e execução de ambientes em SMA. No *CArtAgO* o ambiente é projetado como um conjunto dinâmico de entidades computacionais chamadas de *artefatos*, que representam os serviços e ferramentas que os agentes são capazes de explorar em tempo de execução [5]. O conjunto de artefatos é organizado em um ou múltiplos *workspaces*, que podem ser distribuídos em vários nós de uma rede. Agentes podem entrar e sair dos *workspaces* e, dentro destes, artefatos podem ser criados e descartados dinamicamente pelos agentes [12]. Neste projeto não foi criado nenhum artefato novo, apenas se fez uso dos artefatos organizacionais fornecidos pelo *Moise*.

Moise é um modelo organizacional que decompõe a especificação da organização em três dimensões: estrutural, funcional e deôntica [4]. O modelo permite que o projetista especifique explicitamente a organização do SMA e suas restrições, além de também poder ser usado pelos agentes para raciocinar sobre sua organização. A especificação organizacional do time desenvolvido é descrita a seguir.

B. Estratégia

Podemos definir a estratégia adotada como uma combinação da estratégia organizacional, das estratégias específicas de cada papel, e da estratégia de coordenação.

1) Estratégia Organizacional: A estratégia principal do time foi dividir os agentes em três subgrupos: dois grupos responsáveis por ocupar as melhores zonas do mapa (subgrupos `zona1` e `zona2`), e um encarregado de sabotar o time inimigo (subgrupo `sabotagem`). Para organizar os agentes dessa maneira, foram utilizados os artefatos organizacionais fornecidos pelo *Moise*, e definidas as especificações estrutural (EE), funcional (EF) e deôntica (ED) do SMA.

A EE define os papéis, relações entre papéis, e grupos de uma organização. Cada agente pode assumir um ou mais papéis, e cada papel está relacionado com um conjunto de restrições comportamentais que um agente aceita ao entrar em um grupo. Como mostrado na Figura 3, na EE do *LTI-USP Team* foram definidos os três subgrupos citados anteriormente, e sete possíveis papéis: os cinco papéis especificados pelo cenário (explorador, sabotador, sentinela, reparador e inspetor), e mais dois papéis chamados *"marciano"* e *"coordenador"*. O coordenador é o responsável por conduzir os agentes a ocuparem as melhores zonas de Marte e, diferente dos outros agentes, o coordenador não se comunica com o servidor do MAPC. O marciano é o papel padrão adotado pelos outros agentes no início de uma partida, enquanto não recebem do servidor a informação sobre qual papel desempenhar.

A EF define como os objetivos globais devem ser alcançados, isto é, como esses objetivos são decompostos (em submetas) e distribuídos aos agentes (em missões). Ela também

[9]Disponível em http://sourceforge.net/projects/apleis/.

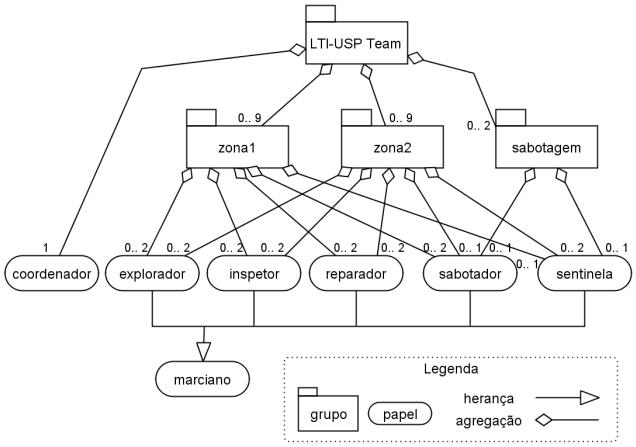

Figura 3. Especificação Estrutural do *"LTI-USP Team"*.

especifica como as sub-metas estão relacionadas, se devem ser alcançadas em paralelo ou em uma certa sequência. Desta forma, cada subgrupo anteriormente especificado possui um meta global associada, definida na EF. Na Figura 4, essas metas globais são representadas como raiz das árvores, que apresentam em suas folhas as sub-metas a serem atingidas pelos agentes. O rótulo que aparece logo acima de uma meta representa a missão ao qual o agente deve estar comprometido, a fim de atingir a meta relacionada. Para cada uma das missões existe um plano que o agente deve seguir.

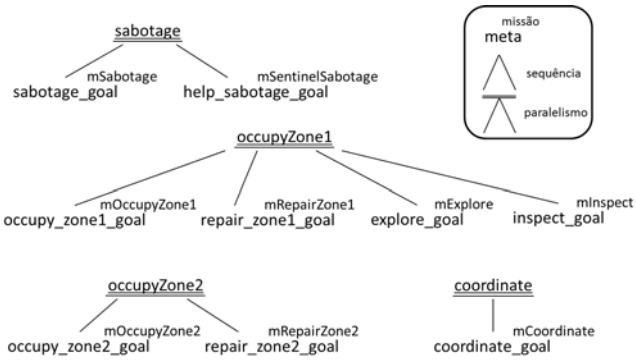

Figura 4. Especificação Funcional do *"LTI-USP Team"*.

As diferentes metas relacionadas a cada subgrupo são apresentadas a seguir:

- `occupyZone1`: Os agentes no grupo `zona1` devem ocupar a melhor zona no grafo seguindo as orientações dadas pelo agente coordenador. Além disso, um dos exploradores fica responsável por sondar os vértices do grafo, a fim de encontrar as melhores zonas a serem ocupadas. Por fim, um inspetor tem a missão de identificar o papel de cada agente no time inimigo. Apenas ao tomar conhecimento do papel de todos os oponentes é que o inspetor se junta ao resto do time na missão de ocupar a melhor zona do grafo.
- `occupyZone2`: Todos os agentes no grupo `zona2` tem a missão de ocupar a segunda melhor zona no grafo, ou ajudar o grupo `zona1` a formar uma área maior. Se este grupo deve ou não se juntar ao outro grupo é determinado pelo coordenador, que verifica a existência ou não de duas boas zonas dispersas no grafo.
- `sabotage`: O grupo `sabotagem` é formado por um sabotador e um sentinela. A missão do sabotador é atacar os oponentes que ocupam bons vértices; e a missão do sentinela é de ajudar na sabotagem, movendo-se para dentro da zona do time inimigo de forma a quebrá-la ou fazê-la diminuir.

Por fim, a ED liga as especificações estrutural e funcional definindo com quais missões um papel tem a obrigação ou a permissão de se comprometer. A ED do *LTI-USP Team* é representada na Tabela II.

Tabela II. ESPECIFICAÇÃO DEÔNTICA DO *"LTI-USP Team"*.

Papel	Missão	Relação Deôntica
explorador	mExplore, mOccupyZone1	permissão
explorador	mOccupyZone2	obrigação
reparador	mRepairZone1, mRepairZone2	obrigação
sabotador	mSabotage, mOccupyZone1, mOccupyZone2	obrigação
sentinela	mSentinelSabotage, mOccupyZone1, mOccupyZone2	obrigação
inspetor	mInspect, mOccupyZone1	permissão
inspetor	mOccupyZone2	obrigação
coordenador	mCoordinate	obrigação

2) Estratégias Dependentes do Papel: Para cada papel, foram desenvolvidas estratégias específicas. Por exemplo, os exploradores devem sondar e examinar todo vértice ou aresta no seu caminho, enquanto os inspetores devem sempre procurar inspecionar inimigos próximos.

Os agentes também se comportam diferentemente, dependendo do papel que desempenham, quando encontram um oponente no mesmo vértice. Os sabotadores sempre atacam o oponente, enquanto os sentinelas procuram se defender e os outros agentes fogem para um outro vértice.

3) Estratégia de Coordenação: A estratégia adotada é baseada na centralização da coordenação, isto é, um único agente (coordenador) é responsável por determinar quais as melhores zonas do mapa e, em seguida, conduzir os outros agentes a ocuparem essas zonas. A escolha da coordenação centralizada foi feita para permitir o rápido desenvolvimento da equipe, uma vez que a motivação principal era se concentrar no uso da plataforma *JaCaMo* e não nos aspectos de coordenação.

Desta forma, o coordenador constrói o seu modelo de mundo através das percepções difundidas pelos outros agentes e, sempre que o modelo de mundo é atualizado, calcula quais são as duas melhores zonas no grafo. O coordenador então pede para que os agentes (de cada um dos dois grupos responsáveis por ocuparem as zonas) se movam para dentro das zonas calculadas. Quando todos os agentes estiverem dentro das zonas, o coordenador passa então a olhar nos vértices vizinhos a estas, procurando posições para as quais os agentes possam se mover a fim de aumentar o tamanho da área conquistada.

IV. ANÁLISE DE DESEMPENHO

O grande obstáculo no desenvolvimento do time foi lidar com os problemas de desempenho relacionados com o uso concorrente dos artefatos organizacionais.

No início de cada partida, os agentes recebem quase que ao mesmo tempo do servidor do MAPC os papéis que irão desempenhar naquela partida para então tentar entrar em um dos três grupos especificados (`zona1`, `zona2` e `sabotagem`). Neste momento, é possível que, por exemplo, todos os quatro sabotadores queiram entrar no grupo `sabotagem`, mas, como mostrado na Figura 3, neste grupo só é permitido um sabotador. Neste caso, três sabotadores serão impedidos de entrar no grupo `sabotagem` e terão que tentar se juntar a outro grupo. Em testes realizados antes da competição, foi notado que a chamada de uma ação organizacional, tais como $adoptRole$ (para a adoção de um papel em um grupo) ou $commitMission$ (para se comprometer a uma missão), é muito cara, e que o número de tentativas feitas por um agente até finalmente conseguir entrar em um grupo podia ser muito alto. Isto fazia com que alguns agentes perdessem alguns passos no início da simulação, já que em cada passo os agentes tem um prazo limitado (1000 ms) para enviarem as suas ações.

Para corrigir esse problema, passamos para o agente coordenador a responsabilidade de distribuir entre os outros agentes os grupos e missões. Desta forma, foi evitada a ocorrência de colisões, como agentes tentando entrar no mesmo grupo ou se comprometer com a mesma missão. Isto melhorou o desempenho do SMA, mas mesmo assim foi possível observar durante a competição que alguns agentes continuavam perdendo passos antes de finalmente conseguir se comprometer com uma missão na organização.

Assim, após o término da competição foram realizados alguns experimentos para avaliar de forma preliminar o desempenho da operação $adoptRole$, em função do número de agentes tentando acessar a operação em concorrência.

No primeiro experimento, foram coletados para cada agente o tempo (em milissegundos) entre a chamada da operação $adoptRole$ e a atualização da base de crenças do agente com o papel adotado. Neste experimento, cada agente tenta assumir um papel diferente na organização, o que significa que não há risco da operação $adoptRole$ falhar. Para cada número de agentes (10, 20, 50 e 100), o experimento foi rodado 20 vezes, e o tempo médio de resposta para um agente adotar um papel é apresentado na Figura 5. É possível notar que o tempo de resposta da operação aumenta consideravelmente com o número de agentes acessando o artefato ao mesmo tempo.

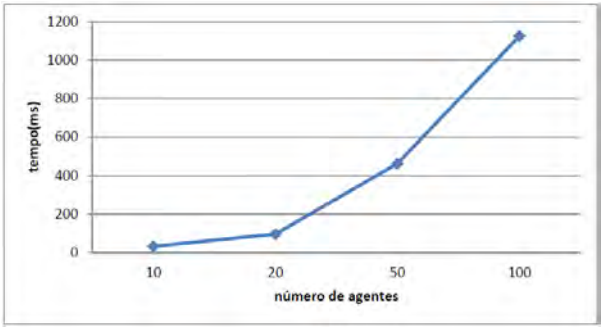

Figura 5. Tempo médio gasto para cada agente adotar um papel, em relação ao número de agentes executando a operação $adoptRole$ em concorrência.

Em outro experimento, novamente o número de papéis é igual ao número de agentes, mas inicialmente todos os agentes tentam adotar o mesmo papel. Como apenas um agente pode adotar cada papel, ocorre que a grande maioria dos agentes não consegue adotar o papel inicial, e então tentam adotar um segundo papel, e assim sucessivamente, até que todos os agentes tenham conseguido um papel na organização. Foram coletados neste experimento, o tempo médio gasto para cada agente entre a primeira chamada da operação $adoptRole$, e a atualização da base de crenças do agente com o papel adotado. Além disso, também foram coletados o número médio de tentativas feita por cada agente até finalmente ter sucesso em adotar um papel. Os resultados obtidos são mostrados na Figura 6.

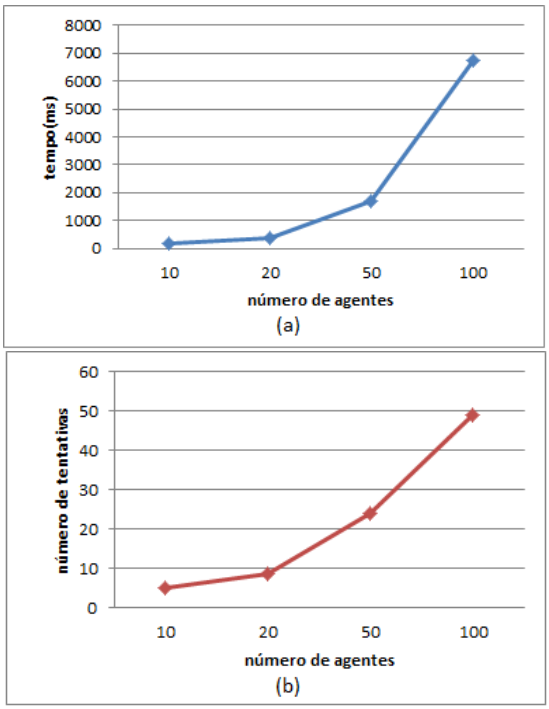

Figura 6. (a) Tempo médio gasto por cada agente para adotar um papel, em relação ao número de agentes executando a operação $adoptRole$ em concorrência; e (b) Número médio de chamadas a operação $adoptRole$ por agente.

Comparando os dois experimentos, é possível notar que o resultado obtido no primeiro, em que cada agente adota um papel diferente, é melhor do que os resultados obtidos no segundo experimento, em que inicialmente os agentes tentam adotar o mesmo papel. Isto reforça a nossa escolha de ter dado ao agente *coordenador* a responsabilidade de distribuir os papéis entre os outros agentes, a fim de reduzir o número de chamadas para a operação $adoptRole$, e assim melhorar o desempenho do SMA.

Além disso, é possível notar que o tempo aumenta consideravelmente com o número de agentes, principalmente para os valores acima de 20 agentes. Esse aumento no tempo de resposta da operação pode ser explicado pela arquitetura utilizada pelo CArtAgO, na qual cada *workspace* possui uma fila para onde vão as requisições dos agentes para a execução das operações nos artefatos, e de onde 20 threads ficam responsáveis por capturar as requisições e executá-las. Assim, quando se tem um número de requisições concorrentes maior

que o número de threads trabalhando para executá-las, é normal que as requisições "em excesso" tenham que ficar esperando mais tempo na fila até serem executadas.

V. Conclusões

A participação de nosso time no MAPC 2012 foi de grande valia no objetivo de conhecer melhor o funcionamento do arcabouço *JaCaMo*. Apesar dos esforços não terem sido direcionados na procura de uma melhor estratégia, o time desenvolvido acabou o torneio em quarto lugar em um total de sete equipes.

Nosso grande obstáculo no desenvolvimento do time foi lidar com os problemas de desempenho relacionados com o uso concorrente dos artefatos organizacionais. Em cenários onde o tempo é limitado, como o enfrentado nessa competição, o desempenho de uma plataforma é um fator muito importante. Assim, os problemas de desempenho encontrados podem acabar dificultando a adoção do *JaCaMo* em tais cenários. Consequentemente, como trabalho futuro pretende-se realizar uma avaliação completa do desempenho e gargalos no *JaCaMo*.

Apesar desses problemas de desempenho, o arcabouço *JaCaMo* provou ser bastante completo, fornecendo todas as ferramentas que precisávamos para desenvolver o SMA. Além disso, o uso do modelo organizacional *Moise* ajudou no projeto da equipe, uma vez que fornece uma forma de estruturar o SMA, não somente através da especificação de grupos e papéis, mas principalmente na definição de como os objetivos globais devem ser alcançados.

Agradecimentos

Jaime Simão Sichman é parcialmente financiado pelo CNPq e FAPESP/Brasil.

Referências

[1] J. Ferber, O. Gutknecht, and F. Michel, "From agents to organizations: an organizational view of multi-agent systems," *Agent-Oriented Software Engineering IV*, no. July 2003, pp. 214–230, 2004.

[2] O. Boissier, J. Hübner, and J. Sichman, "Organization oriented programming: From closed to open organizations," in *Engineering Societies in the Agents World VII*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, vol. 4457, pp. 86–105.

[3] T. Behrens, M. Köster, and F. Schlesinger, "The multi-agent programming contest 2011: a résumé," *Programming Multi-Agent Systems*, pp. 155–172, 2012.

[4] J. Hübner, J. Sichman, and O. Boissier, "Developing organised multiagent systems using the MOISE+ model: programming issues at the system and agent levels," *International Journal of Agent-Oriented Software Engineering*, pp. 1–27, 2007.

[5] O. Boissier, R. H. Bordini, J. F. Hübner, A. Ricci, and A. Santi, "Multi-agent oriented programming with JaCaMo," *Science of Computer Programming*, 2011.

[6] R. Bordini, J. Hübner, and M. Wooldridge, *Programming multi-agent systems in AgentSpeak using Jason*, 2007.

[7] A. Ricci, M. Piunti, and M. Viroli, "Environment programming in multi-agent systems: an artifact-based perspective," *Autonomous Agents and Multi-Agent Systems*, vol. 23, no. 2, pp. 158–192, Jun. 2010.

[8] M. Bratman, *Intentions, Plans, and Practical Reason*. Harvard University Press, 1987.

[9] A. S. Rao, "Agentspeak(l): Bdi agents speak out in a logical computable language," in *Proceedings of the 7th European workshop on Modelling autonomous agents in a multi-agent world : Agents Breaking Away*, ser. MAAMAW '96. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1996, pp. 42–55.

[10] T. M. Behrens, J. Dix, and K. V. Hindriks, "The Environment Interface Standard for Agent-Oriented Programming - Platform Integration Guide and Interface Implementation Guide," *Department of Informatics, Clausthal University of Technology, Technical Report*, vol. IfI-09-10, 2009.

[11] A. Omicini, A. Ricci, and M. Viroli, "Artifacts in the a&a meta-model for multi-agent systems," *Autonomous Agents and Multi-Agent Systems*, vol. 17, no. 3, pp. 432–456, Dec. 2008.

[12] A. Ricci, M. Viroli, and A. Omicini, "CArtAgO: An infrastructure for engineering computational environments in MAS," *3rd Inter. Workshop "Environments for Multi-Agent Systems"(E4MAS)*, 2006.