Multiagent System to search and contracting Tourism services

João Ferreira de Santanna Filho, Scheila Nair Costa, Camila Pontes B. da Costa,
Charbel Szymanski, João Eduardo Hornburg
PPGEAS – Department of Automation and Systems
Universidade Federal de Santa Catarina
Florianópolis, Brasil
{joaosantanna,Camila,charbel}@das.ufsc.br, {nc.scheila, joao.hornburg}@gmail.com

***Abstract*— This paper presents a multi-agent system (MAS) to integrate the service offerings of numerous companies of the tourism segment. This idea was motivated by world sporting events that will happen in the coming years in Brazil (FIFA World Cup and Olympics). The paper demonstrates that MAS's can be a platform for integrating distributed systems resulting in more flexible systems with embedded intelligence.**

***Keywords*— multi-agent system ; distributed systems; Tourism services.**

I. INTRODUÇÃO

Nos anos de 2014 e 2016 o Brasil será sede de dois grandes eventos esportivos de nível mundial, a Copa do mundo de Futebol e as Olimpíadas. De acordo com o SEBRAE [1], a realização desses eventos proporciona uma chance de promoção e atração de investimentos, alavancando significativamente a economia brasileira. Estima-se que o fluxo seja em torno de 600 mil turistas internacionais e 3.100 mil nacionais. É necessário que o setor de serviços se prepare para esse aumento de demanda, em especial o setor Hoteleiro.

Todo esse montante de turistas deve utilizar algum meio para programar e contratar serviços relacionados às viagens. Nesse contexto, a utilização da Internet para o planejamento e compra de pacotes de viagens tem se mostrado uma opção cada vez mais popular. Várias empresas já dispõem de portais para contratação de serviços tais como hotéis, locadoras de carros, restaurantes, bares, serviços de *concierge*, etc. Todavia, esses serviços encontram-se distribuídos cada qual com seu próprio sitio na internet. Dessa maneira, o cliente se vê obrigado a fazer uma pesquisa extensa na internet em busca de melhores preços e melhores ofertas.

Segundo Zagheni and Luna [2], no Brasil, empresas privadas e também do governo não utilizam todo o potencial que a Internet oferece, ou seja, ela é vista apenas como um canal de divulgação, podendo ser comparada com uma versão eletrônica das listas telefônicas. Para que os clientes possam usufruir dos benefícios das novas tecnologias, é necessário que as empresas busquem formas mais simples, ágeis e eficientes de tentar integrar esses serviços, facilitando a busca dos clientes.

A tecnologia de Sistemas Multiagentes (SMA) pode surgir como uma solução para o cenário apresentado. No presente artigo se desenvolve um sistema multiagente (SMA) para a contratação de serviços turísticos. Os agentes representam os vários atores envolvidos no cenário proposto.

Por motivos de ordem prática, o SMA abordou apenas os serviços de Hotelaria e Locação de Automóveis. Porém, visando explorar os diversos recursos disponíveis em Sistemas Multiagentes, foram utilizadas as tecnologias de ontologias, sistemas especialistas e agentes do tipo reativo e cognitivo (Agentes BDI). Para a comunicação entre os agentes foram utilizados dois protocolos diferentes de transação, um baseado em barganha (utilizado pelos agentes hotéis) e o *Contract Net*, utilizado pelas locadoras de carros. O uso de todas essas tecnologias em um único sistema visa demonstrar que um SMA desse tipo pode atender a requisitos diferentes demandados pelas empresas prestadoras de serviço.

O objetivo principal deste trabalho é demonstrar a viabilidade de um SMA agregar serviços distribuídos pela internet visando oferecer uma ferramenta de busca e contratação centralizada que englobe vários serviços.

II. DEFINIÇÃO DO ESCOPO DO SMA

A partir da motivação apresentada, foram implementados no SMA agentes para representar os serviços de Hotelaria, os serviços de locação de automóveis e agentes intermediadores de negociação.

Fig. 1. : Escopo do SMA.

O escopo deste trabalho foi definido como ilustrado na Figura 1. Um Agente de Viagens interage com um agente Gestor de Hotel para encontrar agentes hotéis, e com um agente Gestor de Locadora de Automóveis para encontrar agentes locadoras de Automóveis.

Cada um dos elementos presentes na Figura 1 é implementado na prática por um agente. Dessa forma, temos: 1 agente de viagens, 1 agente gestor de Hotéis, 1 agente gestor de locadoras, *n* agentes de hotéis e *n* agentes de locadoras de automóveis.

O cliente faz algumas escolhas quanto ao serviço. Com base nessas escolhas, o agente classifica o tipo de Hotel e o tipo de automóvel. Depois dessa classificação, o agente de viagens requisita ajuda a outros dois agentes (Gestor de Hotel e Gestor de Locadora de Automóveis) para realizar buscas por hotéis e automóveis segundo as escolhas feitas pelo cliente.

Por fim, será apresentado ao cliente um pacote contendo: a reserva de hospedagem em um hotel e uma locadora com automóvel reservado.

III. CONSIDERAÇÕES SOBRE A ESCOLHA DAS FERRAMENTAS DE IMPLEMENTAÇÃO DOS AGENTES

A plataforma JADE (*Java Agent Development Framework*) foi escolhida como base do SMA e grande parte dos agentes que compõe o SMA foi implementado usando esse *middleware*. Segundo Bellifemine, et al. [3], Jade é um *middleware* que facilita o desenvolvimento de SMA's e já vem sendo usado com sucesso em aplicações de diversos setores como gestão de cadeia de suprimentos, gestão de frotas, leilões e turismo.

Algumas características levaram à opção pelo uso do JADE, principalmente o total suporte ao padrão FIPA (*Foundation for Intelligent Physical Agents*) para a comunicação entre os agentes [3]. O uso desse padrão de comunicação permitiu que, em um segundo momento, os agentes JADE pudessem conversar com os agentes Jason (*A Java-based AgentSpeak Interpreter Used with Saci For Multi-Agent Distribution Over the Net*) [4] utilizando o JADE como plataforma de comunicação. Outro fator que levou à adoção da plataforma JADE no projeto foi o fato dele ser implementado na linguagem de programação Java. O *middleware* foi desenvolvido para prover um conjunto de API's que são independentes do tipo de rede e da versão de Java utilizada. Dessa forma, o SMA pode ser facilmente estendido no futuro e ter agentes rodando a partir de dispositivos móveis como celulares que suportem a plataforma Java.

Além do JADE, parte do sistema foi desenvolvido usando Jason. A motivação para o uso do interpretador Jason foi poder utilizar agentes cognitivos baseados na arquitetura BDI (B*elief, Desires and Intentions*) [5]. Diferentes dos agentes JADE, que são reativos, os agentes Jason possuem crenças, objetivos, planos e intenções, sendo a programação realizada na linguagem AgentSpeak, e a comunicação entre agentes baseada na teoria de atos de fala[6]. Em agentes Jason os planos executados e as crenças podem mudar dinamicamente conforme os agentes vão interagindo no SMA. Dessa forma, um agente que esteja participando de uma negociação pode facilmente mudar de estratégia, desde que suas crenças sejam alteradas. Apesar dos agentes Jason apresentarem uma arquitetura mais rica que os agentes JADE, no presente trabalho eles só foram utilizados para negociação do tipo *Contract NET*, onde cada agente Jason representava uma locadora de automóveis. No presente artigo só integramos agentes com arquiteturas diferentes em um mesmo SMA. As características de agentes BDI podem ser melhor exploradas em um futuro trabalho.

IV. DESENVOLVIMENTO DO SMA

Atualmente existe um grande número de metodologias na área de Sistemas Multiagentes. Segundo Akbari [7], de 1990 até 2010 foram desenvolvidas cerca de 75 metodologias diferentes. A metodologia selecionada para a modelagem do sistema foi a metodologia "Tropos", em razão da mesma estar bem documentada e apresentar ferramentas computacionais de apoio na plataforma em que o SMA estava sendo desenvolvido.

Tropos é uma metodologia de desenvolvimento de *software* orientada a agentes. Foi originalmente desenvolvida na Universidade de Toronto, no Canadá. Tropos tem como foco principal a análise de requisitos e adota o paradigma i* Mylopoulos, et al. [8] como base. O i* oferece conceitos como atores, objetivos e dependências destinadas para modelar as estruturas sociais e descrever relações entre eles. A metodologia Tropos é dividida em 5 fases de desenvolvimento: (1) requisitos iniciais; (2) requisitos finais; (3) projeto arquitetural; (4) projeto detalhado e (5) implementação.

A. Requisitos iniciais do SMA

Utilizando o escopo definido, podemos identificar as partes interessadas, suas intenções (*goals*) e seus relacionamentos. Além dos agentes especificados no escopo, foi criado um novo agente categorizador de hotéis. A tarefa desse agente é receber as características escolhidas pelos clientes e, a partir dessas características, ranquear o hotel em uma categoria: 3, 4 ou 5 estrelas.

Após definidos os interessados e suas intenções, uma nova análise foi realizada para identificar quais planos os atores deverão realizar para atingir suas intenções. Também foram identificados os relacionamentos entre os atores para a realização dos planos.

Utilizando a ferramenta TAOM4E [9] , as intenções foram transcritas como metas ou meta-soft. Temos ainda, a partir da análise inicial, os planos e as dependências entre os atores, além dos recursos identificados que foram gerados por eles.

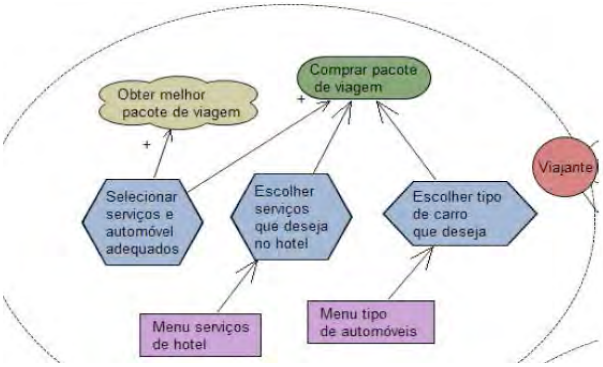

Fig. 2. Diagrama de metas do ator viajante.

Depois de feita a modelagem do Diagrama de Atores, foi modelado o Diagrama de Metas. Essa tarefa consiste em verificar a possibilidade de decompor cada meta que um ator possui em sub-metas, podendo surgir novas metas e novos planos. Depois foi realizada a análise meio-fim, definindo quais planos são os meios para alcançar as metas. A Figura 2 apresenta o diagrama de metas do ator viajante.

B. Requisitos Finais do SMA

O primeiro passo nessa fase foi inserir um ator que representa a interface com o usuário. Esse ator foi denominado SAV (Sistema de Apoio a Viagens). Depois realizou-se uma nova análise em relação as intenções, planos e relacionamentos, incluindo esse ator. O resultado pode ser conferido na Tabela 1.

TABELA I. INTENÇÕES, PLANOS E RELACIONAMENTO ENTRE ATORES

Ator	Intenções (*goals*)	Plano	Relacionamentos
Viajante	- Comprar pacote de viagem - Obter o melhor Pacote de viagem	- Escolher serviço que deseja no hotel - Escolher tipo de carro que deseja	SAV
SAV	- Vender pacote de viagem - Categorizar Hotel - Interagir com Viajante	- Requisitar reserva de hotel - Requisitar locação de automóvel - Analisar serviços escolhidos para categorizar hotéis	- Gestor Hotel - Gestor Locadora - Viajante
Gestor Hotel	Fechar negócio com Hotel	Negociar reserva	Hotéis
Gestor Locadora de Automóveis	Fechar negócio com Locadora	Negociar locação	Locadoras
Hotel	- Obter hóspede - Fornecer serviços	Barganhar proposta	Gestor Hotel
Locadora de Automóveis	Obter Locatário	- Registrar serviços - Informar valor	Gestor Locadora

A partir da Tabela 1 foi possível gerar o Diagrama de Atores da fase de Requisitos Finais e, em seguida, realizar a modelagem do diagrama de metas para o ator SAV.

C. Projeto Arquitetural

Segundo Silva [10], existem um conjunto de estilos arquiteturais já definidos para SMA's. Nesse projeto, identificamos que parte da estrutura é Flat e parte é Hierárquica.

Analisando o diagrama de metas do ator SAV, identificamos cinco papéis. Cada papel foi especificado segundo seus objetivos, responsabilidades, colaboradores, habilidades e normas, que são representadas na Tabela 2.

TABELA 2. ESPECIFICAÇÃO DE PAPÉIS

Agente de viagem	Objetivos:	Montar pacote de viagens
	Responsabilidades:	Procurar Gestor de hotel e gestor de locadora, Categorizar Hotel, Requisitar reserva em Hotel e locação de Automóvel, Publicar pacote montado.
	Colaboradores:	Gerenciador de Hotéis e Gerenciador de Locadoras
	Habilidades:	Conhecer os Hotéis selecionados pelo viajante
	Normas:	Acessar a base de serviços
Gestor de Hotéis	Objetivos:	Negociar reserva em um Hotel
	Responsabilidades:	Procurar Hotéis de uma categoria específica em uma lista, Enviar mensagem requisitando preço para Hotéis, Selecionar melhor oferta.
	Colaboradores:	Hotéis e Gerenciador de Viagens (SAV)
	Habilidades:	Conhecer a categoria do hotel procurado e as propostas do hotéis envolvidos na negociação
	Normas:	Acessar a base de serviços
Gestor de Locadoras	Objetivos:	Negociar locação de automóvel
	Responsabilidades:	Enviar mensagens para locadoras de automóveis, Selecionar melhor oferta.
	Colaboradores:	Locadoras de automóveis e Gerenciador de viagens (SAV)
	Habilidades:	Realizar a *Contract Net* com as locadoras encontradas
	Normas:	Acessar a lista de locadoras disponíveis
Hotel	Objetivos:	Vender a reserva em Hotel
	Responsabilidades:	Responder mensagem com valor
	Colaboradores:	Gerenciador de Hotéis
	Habilidades:	Conhecer ou descobrir sua categoria
	Normas:	Conhecer sua categoria. Caso contrário, deve acessar uma ontologia para descobrir em que categoria se encaixa.
Locadora	Objetivos:	Alugar Automóvel
	Responsabilidades:	Se registrar junto a Gerenciador de Locadoras, Responder mensagem com valor do respectivo automóvel.
	Colaboradores:	Gerenciador de Locadoras
	Habilidades:	Conhecer o tipo de carro desejado
	Normas:	Acesso ao tipo de carro desejado

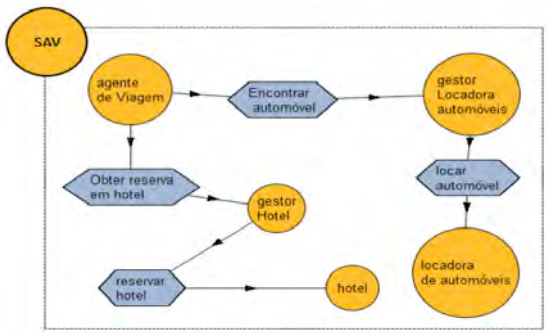

Fig. 3. Diagrama Arquitetural do Sistema.

Na Figura 3 é apresentado o Diagrama Arquitetural do sistema, resultante do mapeamento em conformidade com o estilo arquitetural escolhido.

Finalizou-se a fase de Projeto Arquitetural com a elaboração do diagrama arquitetural e a definição dos papéis.

D. Projeto Detalhado

Nessa fase foi descrita a estrutura interna de cada papel e detalhes de projeto.

Analisando o diagrama de metas do ator SAV (Figura 3), identificamos cinco papéis. Cada papel foi especificado segundo seus objetivos, responsabilidades, colaboradores, habilidades e normas, que são representadas na Tabela 2.

1) Funcionamento do protocolo de Barganha (Gestor Hotel X Hotel)

O protocolo de negociação implementado funciona utilizando barganha em 4 rodadas de negociação. No primeiro *round* o agente Gestor de Hotéis recebe uma requisição para buscar por um hotel com determinada categoria (3, 4 ou 5 estrelas). Baseado nessa requisição, o gestor consulta o serviço de páginas amarelas e recebe uma lista com agentes que representam os hotéis da categoria procurada. Usando a lista, o agente gestor envia pedidos para todos os hotéis utilizando a performativa *request* com o seguinte conteúdo na mensagem (1, 2 ,0). O 1 informa aos agentes que eles estão participando do protocolo de barganha e que estão na primeira rodada de negociação. O 2 representa que a busca é para quartos da 2ª semana do ano. No projeto do protocolo definimos o segundo campo como uma informação de data, mas essa informação não foi utilizada para tomada de decisão. Se assumiu que na data escolhida todos os Hotéis terão vaga. O 0 (zero) da mensagem é o campo que indica a melhor oferta atual. Como os agentes ainda não ofertaram preços, se envia preço 0 como indicador da primeira rodada. Os agentes recebem essa mensagem e, de acordo com suas próprias estratégias, ofertam um valor para o serviço.

Na segunda rodada o agente gestor processa as ofertas que chegam e decide como melhor preço a menor oferta enviada pelos agentes. A seguir o gestor envia uma mensagem com performativa *inform* com o seguinte conteúdo (2, 2, ***melhor Preço***). Como no primeiro envio, o primeiro campo de valor 2 representa que estamos na 2º rodada de barganha, o segundo 2 continua representando a semana de locação do quarto, e o último campo carrega o melhor preço escolhido do conjunto de preços da rodada anterior. Genericamente podemos descrever que o conteúdo das mensagens segue o seguinte formato: **(*n*,*x*,*y*)**, sendo: *n* (número da rodada), *x* (campo de data para reserva do quarto) e *y* (menor preço da última rodada de negociação).

O processo continua até a 4ª rodada. Na rodada final o agente gestor finaliza a negociação enviando uma mensagem com a performativa *Accept Proposal* para o agente do hotel vencedor que deve, nesse momento, reservar o quarto. O restante dos hotéis não recebe mensagem nenhuma e, como o sistema foi projetado para que os agentes dos hotéis não reservem quartos antes da 4ª rodada, não ocorre prejuízo para os hotéis perdedores.

Os agentes do SMA relacionados a hotéis foram todos feitos usando a plataforma JADE[11]. O agente Gestor de Hotéis foi projetado para controlar o protocolo de barganha utilizando um *Tick Behaviour* temporizado para 6 segundos. Nesse meio tempo existe um *Cyclic Behaviour* que atualiza a lista de hotéis continuamente. O agente gestor envia mensagens para esse conjunto atualizado de agentes. Dessa forma, o protocolo permite que um hotel que não participou na 1ª rodada possa participar da 2ª rodada ou da 3ª. Outra escolha de projeto foi que os agentes hotéis não respondam quando não puderem ofertar um preço melhor que o barganhado. Isso foi feito para diminuir a quantidade de mensagens circulando na rede.

2) *Estratégia dos Agentes Hotéis*

Como escolha de projeto desenhamos três agentes com estratégias distintas para participar da barganha de preços. Para essas três classes de agentes com comportamentos distintos foram projetadas mais três classes dessas para cada categoria de hotel (3, 4 e 5 estrelas) com diferentes preços do serviço segundo a categoria dos hotéis . Os três tipos de hotéis são os seguintes:

- Hotel Escada: Oferta 3 preços fixos, um para cada rodada sem variar.
- Hotel Turco: Oferta um preço randômico próximo ao primeiro preço do hotel escada e, a partir daí, nas outras rodadas, sempre que receber do agente Gestor de Hotéis um preço, diminui em R$ 1 (um real, moeda brasileira) a melhor oferta e envia essa oferta ao gestor.
- Hotel Randômico: Como o turco, sua primeira oferta é gerada por um número randômico próximo ao do hotel escada. A partir daí, nas próximas rodadas ele recebe o melhor preço, gera um número randômico e diminui do melhor preço e faz sua oferta com o resultado.

A partir desses hotéis são criados tipos específicos para cada categoria. Cada um desses hotéis com seu próprio agente. Dessa forma temos: Hotel Escada 3, 4 e 5 estrelas; Hotel Turco 3, 4 e 5 estrelas; Hotel Randômico 3, 4 e 5 estrelas. A única alteração feita entre esses hotéis é a precificação do serviço em cada *round*.

3) *Uso de ontologias para um Hotel que não sabe a que classe pertence*

Foi propositadamente desenvolvido no SMA um agente Hotel que não tem conhecimento da sua categoria e utiliza a ontologia criada para descobrir em que categoria se encaixa. Esse agente foi criado prevendo a participação de estabelecimentos não classificados previamente, mas que poderiam participar da negociação. Dessa forma, foi definida uma ontologia para representar os atributos de um hotel. Esta ontologia define as classes *Service*, Hotel e *Room*. A classe *Service* possui como subclasses os serviços que um hotel pode oferecer. A classe Hotel possui como subclasses as categorias de hotéis (“estrelagem”). A classe *Room* define o quarto de um hotel.

Utilizando a ferramenta Protégé[12], temos a hierarquia completa de todas as classes. Foram definidas algumas propriedades para cada classe, tais como: *hasRoom, isRoomOf, hasService e isServiceOf.* A propriedade *hasRoom* possui como domínio a classe Hotel e como *range* a classe *Room.* A propriedade *isRoomOf* é a inversa de *hasRoom.* A propriedade *hasService* também possui como domínio a classe Hotel, e possui como *range* a classe *Service.* Já *isServiceOf* é a propriedade inversa de *hasService*. Além

disso, as subclasses de Hotel possuem restrições (superclasses anônimas), na forma de *hasService some xxx*, que associam cada categoria de hotel aos serviços que eles oferecem.

O agente hotel que utiliza a ontologia inicializa aleatoriamente uma lista de serviços, simulando um hotel não ranqueado tentando participar da negociação. A seguir, o agente carrega a ontologia e a partir da sua lista de serviços realiza um *reasoning* para determinar qual é a sua categoria (3Star, 4Star ou 5Star). O agente também pode concluir que o conjunto de serviços por ele ofertados não atendem os requisitos de nenhuma das categorias. Se o agente se enquadra em alguma categoria, ele se registra nas páginas amarelas para aquela categoria, caso contrário, ele não se registra e não participa de negociações, sua estratégia de negociação é igual à do Hotel Escada.

4) *Sistema Especialista*

Um sistema especialista embutido no agente que interage com usuário (SAV) foi desenvolvido utilizando a ferramenta Jess [13] para fazer a classificação do tipo de hotel que o usuário deseja a partir dos serviços escolhidos. Dessa forma, o agente pode requisitar ao Gestor de Hotéis um hotel com uma classificação com relação às estrelas (3, 4 ou 5 estrelas).

Para definir os requisitos básicos de cada categoria de hotel, foram utilizadas informações do Sistema Brasileiro de Classificação de Meios de Hospedagem [14] A partir dessa classificação, selecionamos alguns serviços que hotéis devem possuir para obter determinada classificação.

Para inserir como um fato os serviços básicos de cada hotel no sistema especialista, foi usada a palavra-chave *deffacts*, que insere os fatos quando é dado um (reset) no programa, conforme mostra o código exemplo a seguir:

(deffacts hoteis)
(hotel3 Internet Frigobar Estacionamento Cafe)
(hotel4 Internet Frigobar Estacionamento Cafe CafeQuarto ServicoQuarto Manobrista)
(hotel5 Internet Frigobar Estacionamento Cafe CafeQuarto ServicoQuarto Manobrista SalaoEventos Banheira Concierge)

O sistema de viagem executa o sistema especialista acrescentando um fato tipo lista com cabeçalho “serviços”com os serviços solicitados pelo usuário. Por exemplo, se um usuário solicitou Banheira e Internet, o Sistema de Viagem acrescentaria o seguinte fato no SE: (serviços Banheira Internet).

5) Funcionamento do agente gestor de locadora de Automóveis

Utilizando o Jason, foi criado um agente BDI do tipo Gestor que utiliza uma Rede de Contrato (*Contract Net*) como suporte para a coordenação da negociação, que tem como estratégia escolher a empresa que oferece a menor diária para o carro solicitado pelo cliente.

Como crença inicial, o agente tem que todas as respostas foram recebidas se o número de participantes for igual ao número de propostas somado ao número de recusas. O agente Gestor de Locadora possui como objetivo inicial se registrar nas páginas amarelas, para que o sistema de viagem possa encontrá-lo. Para atingir esse objetivo, ele tem um plano que chama uma ação interna “jadedf.register”, que o registra como provedor de serviços do tipo “locação-carros” e com o nome “Gestor-LocadoraCarros”. Existe um segundo plano para o caso de ter sido adicionado à sua base de crenças o recebimento de uma mensagem KQML do tipo CFP (*Call for Proposal*). Nesse plano é armazenado como uma crença o remetente da mensagem (agente que solicitou um carro) e então é adicionado um novo objetivo, o de iniciar a rede de contrato *(“!iniciaContractNet”).*

Para atingir o objetivo de iniciar a rede de contrato, existe um plano de esperar 2 segundos para os participantes se registrarem, guardar o estado da RdC (rede de contrato) como fase de “proposta”, procurar por todos os participantes que se registraram e enviar para cada um deles um pedido de proposta para o carro solicitado na CFP recebida. Em seguida ele espera mais 4 segundos e adiciona o plano de “contratar”. Enquanto isso, o agente gestor recebe as propostas e recusas dos participantes. Quando todas as propostas foram recebidas, ele adiciona o objetivo de “contratar”.

O plano para contratar é executado somente se o estado da RdC atual é “proposta”. Ele consiste de alterar o estado da RdC de “proposta” para “contratar”, encontrar todas as ofertas dos participantes, calcular o menor valor de oferta e então determinar o ofertante desse valor como Agente Vencedor. Então, ele adiciona como novo objetivo anunciar o resultado (avisa ao agente vencedor que ele ganhou e aos outros que a proposta foi recusada) e envia para o remetente a empresa de locação que venceu a RdC, juntamente com sua oferta vencedora. O estado da RdC é alterado para “terminado” e a crença das propostas recebidas e da mensagem KQML daquele remetente são apagadas.

6) Funcionamento dos agentes Locadoras de Automóveis

As locadoras de automóveis foram implementadas como agentes Jason. Foram criados 4 agentes locadoras para fins de simulação, cada um com estoque de carros diferente. Os agentes possuem como crenças iniciais os carros com os quais ele trabalha ou não trabalha, e um preço randômico para ofertar. Os agentes possuem um plano inicial de enviar uma mensagem para o Gestor de Locadora, se registrando para participar das RdC gerenciadas por ele.

Caso um agente tenha uma crença de uma CFP vinda do Gestor solicitando um carro, e ele tenha um carro da categoria disponível no estoque. O agente executa o plano de responder à essa CFP, que reserva um carro do tipo solicitado, adiciona uma crença com a proposta feita e envia a proposta para o Gestor. Se o agente receber uma crença do tipo “aceitar proposta” do Gestor de Locadora, significa que o Gestor está avisando que ele ganhou a RdC. Então ele mantém a reserva do carro para o cliente e apaga as crenças com relação àquela RdC, finalizando a contratação. Caso o agente tenha perdido uma RdC, ele receberá uma crença “recusa proposta” do Gestor de Locadora. Então o agente executará o plano de atualizar o estoque, reinserindo o carro que havia sido reservado, e apagará as crenças com relação àquela RdC.

Finalmente, caso o agente não trabalhe com o carro solicitado ou o carro tenha acabado no estoque, é executado um segundo plano onde o agente envia uma mensagem de recusa para o Gestor, informando que ele não participará da RdC

E. Implementação

O SMA foi implementado na plataforma JADE. Os agentes Gestor de Locadora de Automóveis e Locadora de Automóveis são agentes BDI feitos em Jason. O Gestor de Hotéis e os Hotéis são agentes reativos feitos em Java na plataforma JADE.

O agente SAV foi codificado em Java/JADE [11]. Embutido nesse agente, temos um sistema especialista feito em Jess [15] (anteriormente mencionado) para classificar os hotéis segundo escolhas do cliente.

Fig. 4. Solicitação ao SMA de Hotel e Carro econômico com Ar condicionado

Adicionalmente foi implementado uma GUI (*Graphical User Interface*) usando Java/Swing para servir de interface com o usuário (Figura 4). Essa interface captura os serviços do hotel que o usuário necessita, bem como o tipo de carro, e passa essa lista para o "Agente de Viagens", que por sua vez inicia todo o processo de busca no SMA.

V. CONSIDERAÇÕES FINAIS

O potencial de um Sistema Multiagentes vai muito além do que foi apresentado nesse trabalho. SMA's representam um novo paradigma na área de desenvolvimento de *software* para construção de sistemas distribuídos e na integração de sistemas. O SMA desenvolvido nesse artigo demonstra que tal tecnologia pode ser uma alternativa viável para a construção de uma ferramenta para atender ao cenário proposto.

Por praticidade, os agentes do SMA foram implementados de forma semelhante. Em um sistema real, cada empresa poderia implementar seu(s) agente(s) de acordo com o seu modelo e políticas de negócio, usando diversos protocolos de negociação em um mesmo agente. Dessa forma, um agente poderia participar simultaneamente em negociações do tipo barganha, *Contract Net* e leilão, bastando que fosse programado para isso. Nesse caso seria essencial o seguinte: os agentes intermediários (Intermediários na negociação entre o cliente e os prestadores de serviço), aqui representados pelos agentes Gestor de Hotel e Gestor de Locadora, teriam que informar que tipo de protocolo de negociação está sendo utilizado antes de iniciar a negociação propriamente dita. Além disso, tais protocolos teriam que ser padronizados e de conhecimento dos demais agentes. Dessa forma, um agente poderia participar de vários tipos de negociações diferentes, desde que utilizasse os protocolos padronizados.

Para os agentes intermediários também é valido o uso de vários protocolos de negociação. Tais agentes poderiam ser implementados pelas Secretarias de Turismo locais. Um agente desse tipo poderia mudar de protocolo de negociação segundo a demanda por determinado serviço, como por exemplo, utilizar um protocolo que maximize o lucro em época de maior demanda pelo serviço.

Diferente da implementação do SMA apresentada nesse artigo, um único agente intermediário poderia lidar com vários tipos de serviços diferentes, como por exemplo, um mesmo agente intermediário servir para negociações com hotéis, bares e restaurantes, etc. Dessa maneira é possível perceber que a tecnologia de SMA's pode ser utilizada para a construção de SMA's muito mais complexos que o demonstrado no artigo. Porém, deve-se sempre levar em conta que um agente intermediário lida com dezenas de agentes simultaneamente e o fator escala sempre deve ser levado em conta para não sobrecarregar a infraestrutura de comunicação e a capacidade de processamento do agente.

Outra característica importante desse tipo de sistema é que a complexidade de um SMA pode crescer, mas isso fica transparente para os usuários. Um cliente procurando serviços não precisa saber dos detalhes de implementação dos agentes intermediários, basta que os agentes envolvidos utilizem protocolos padrão de negociação e uma mesma ontologia de serviços (mesma definição para nomes e detalhes dos serviços). Essa transparência se estende ao longo do SMA, uma vez que cada um dos atores (Secretaria de Turismo, hotéis, restaurantes, locadoras de automóveis, bares, etc.) vai cuidar da implementação do seu próprio agente sem precisar conhecer os detalhes de funcionamento do restante dos agentes do SMA.

No SMA de exemplo, o sistema ficou centralizado nos dois agentes intermediários (agente Gestor de Hotel e agente Gestor de Locadora de Automóveis), mas não necessariamente precisa ser dessa forma. Um SMA mais completo poderia contar com diversos gestores diferentes representando regiões distintas geograficamente. Dessa maneira seria possível que um agente de busca mais especializado procurasse por um conjunto de serviços e informasse ao cliente em qual região/estado seria mais vantajoso passar as férias. A plataforma JADE permite a comunicação com diversos SMA's distribuídos.

Dentro do cenário proposto nesse artigo, as Secretarias de Turismo poderiam fomentar a criação de um catálogo eletrônico de serviços turísticos que abrangesse todos seus associados. Dessa maneira, agentes poderiam integrar seus serviços em um único/ou vários SMA's de serviços turísticos.

Sistemas mais sofisticados lidam com APIs e tecnologias que abstraem máquinas de inferência, protocolos de interação e estruturas de informações. Algumas delas foram utilizadas nesse trabalho. Porém, tais ferramentas apresentam um conjunto bem maior de funcionalidades habilitando SMA para lidar com cenários bem mais complexos que o apresentado aqui.

REFERENCES

[1] SEBRAE, "Copa 2014 - oportunidades e desafios," 2011.

[2] E. S. d. S. Zagheni and M. M. M. Luna, "Canais de distribuição do turismo e as tecnologias de informação: um panorama da realidade nacional," *Revista Produção Online*, vol. 11, pp. 476-502, 2011.

[3] F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa, "Title," unpublished|.

[4] R. H. BORDINI, J. F. HUBNER, and M. WOOLDRIDGE, *Programming Multi-Agent Systems in AgentSpeak using Jason*. University of Liverpool, UK, 2007.

[5] A. S. Rao and M. P. Georgeff, "An Abstract Architecture for Rational Agents," in *Principles of knowledge representation and reasoning: proceedings of the third international conference (KR'92)*, 1992, p. 439.

[6] R. H. Bordini and R. Vieira, "Linguagens de Programação Orientadas a Agentes: uma introdução baseada em AgentSpeak (L)," *Revista de informática teórica e aplicada. Porto Alegre. Vol. 10, n. 1 (2003), p. 7-38*, 2003.

[7] O. Z. Akbari, "A survey of agent-oriented software engineering paradigm: Towards its industrial acceptance," *J. Comput. Engg. Res*, vol. 1, pp. 14-28, 2010.

[8] J. Mylopoulos, M. Kolp, and J. Castro, "UML for agent-oriented software development: The Tropos proposal," in *≪ UML≫ 2001—The Unified Modeling Language. Modeling Languages, Concepts, and Tools*, ed: Springer, 2001, pp. 422-441.

[9] D. Bertolini, A. Novikau, A. Susi, and A. Perini, "TAOM4E: an Eclipse ready tool for Agent-Oriented Modeling. Issue on the development process," *University of Trento, Trento, Italy*, 2006.

[10] M. Silva, "U-TROPOS: uma proposta de processo unificado para apoiar o desenvolvimento de software orientado a agentes," Dissertação. Recife, 2008-Universidade Federal de Pernambuco, 2008.

[11] F. Bellifemine, G. Caire, and T. Trucco, "Jade Programmer's Guide. Java Agent Development Framework (2010)," ed.

[12] N. F. Noy, M. Crubézy, R. W. Fergerson, H. Knublauch, S. W. Tu, J. Vendetti, *et al.*, "Protege-2000: an open-source ontology-development and knowledge-acquisition environment," in *AMIA Annu Symp Proc*, 2003, p. 953.

[13] E. Friedman-Hill, "Jess, the rule engine for the java platform," ed, 2003.

[14] D. L. Candeia, "Classificação dos meios de hospedagem," 2004.

[15] E. J. Friedman-Hill, "Jess, the java expert system shell," *Distributed Computing Systems, Sandia National Laboratories, USA*, 1997.