Desenvolvimento de um *Middleware* entre a ferramenta SUMO e o *framework* JaCaMo

Alexis v. H. Heijmeijer[1], Gleifer V. Alves[1]

[1]Departamento Acadêmico de Informática – Universidade Tecnológica Federal do Paraná (UTFPR) – Ponta Grossa – PR – Brasil

{aheijmeijer@uol.com.br, gleifer@utfpr.edu.br}

Abstract. *Smart City is a city where its infrastructure works with technology to reduce daily problems. The lack of parking spots is one of these problemas that can be controlled by creating Smart Parking systems. The MAPS (MultiAgent Parking System) project studies and develops multiagent systems for smart parking using JaCaMo framework to implement agents and artifacts. This paper presents the use of SUMO simulation tool in the MAPS project. In order to do so, we have developed a middleware between SUMO and JaCaMo that allows the representation and visualization of the simulation executed by MAPS agents.*

Keywords: *Smart Parking. Urban Simulation. Intelligent Agents. SUMO. JaCaMo Framework. MAPS-SUMO Middleware.*

Resumo. *O conceito de Cidade Inteligente utiliza a infraestrutura de uma cidade para trabalhar de maneira sincronizada com a tecnologia, com o objetivo de reduzir problemas encontrados no dia a dia. Um desses problemas é a falta de vagas em estacionamentos urbanos que pode ser controlado através da implantação de sistemas de Smart Parking (ou estacionamento inteligente). O projeto MAPS (MultiAgent Parking System) busca justamente aplicar sistemas multiagentes no gerenciamento de estacionamentos inteligentes, utilizando o framework JaCaMo para implementação de agentes e artefatos. Este trabalho visa implantar ao projeto MAPS o uso da ferramenta de simulação SUMO. Para tal é construído um middleware de comunicação entre o SUMO e o JaCaMo, o qual permite representar e visualizar a simulação executada pelos agentes do MAPS.*

Palavras-chave: *Smart Parking. Simulação Urbana. Agentes Inteligentes. SUMO. Framework JaCaMo. MAPS-SUMO Middleware.*

1. Introdução

Smart City (em português, cidade inteligente) mostra-se uma realidade no mundo atual devido ao rápido avanço da tecnologia. *Smart City* é definido como uma cidade onde a sua infraestrutura trabalha de maneira sincronizada com a tecnologia, permitindo aos agentes viverem e interagirem nesse ambiente, contribuindo para que a cidade se torne mais eficiente (Batty et al. 2012). Cidades inteligentes contemplam diversos pontos, como segurança pública integrada e vigilância através de câmeras. Outro ponto é a Mobilidade Urbana, que corresponde às necessidades humanas de deslocamento em um ambiente de acordo com as dimensões do espaço e das atividades exercidas nele. Ainda assim, a

mobilidade é um tópico muito abrangente, porém, um dos pontos mais importantes é a questão do trânsito. O trânsito envolve toda a parte de infraestrutura e serviços de transporte, sejam eles públicos ou privados, tais como estacionamento, monitoramento de transporte público, monitoramento de vias, entre outros.

Estacionamento pertence à parte de infraestrutura de trânsito de uma cidade e pode apresentar diversos problemas, como a falta de vagas e até congestionamento dentro e fora dos mesmos. Segundo pesquisa destacada em Koster, Koch e Bazzan (2014), 40% do trânsito de Nova Iorque é causado por carros em busca de vagas de estacionamento. Segundo o Departamento Nacional de Trânsito (DENATRAN), houve um aumento de 130% na frota de veículos em circulação no Brasil entre janeiro de 2005 e janeiro de 2016. Este aumento reflete diretamente na busca de vagas. Por exemplo, o projeto desenvolvido pela Deutsche Telekom em 2014, mostra que 30% do trânsito de grandes cidades é gerado por motoristas em busca de vagas de estacionamento. Este mesmo projeto visa construir uma estrutura inteligente na cidade de Pisa (Itália), também prova que a implantação do conceito de *Smart City* garante um melhor fluxo de trânsito, reduz a emissão de gás carbônico e facilita a procura por vagas de estacionamento. Projetos como este aplicam o conceito de *Smart Parking*, o qual consiste no uso de tecnologias para automatizar, facilitar e viabilizar o melhor uso de espaços de estacionamento.

Revathi e Dhulipala (2012) descrevem diferentes tipos de *Smart Parking*, classificados conforme suas características: sistemas de pagamentos, estacionamentos automatizados, sistemas inteligentes, sistemas de condução a vagas, entre outros. Entre eles destaca-se o *Agent Based Guiding System* (ABGS), em português, Sistema de Condução Baseado em Agentes, o qual caracteriza o tipo de *Smart Parking* utilizado neste trabalho. O ABGS trabalha com agentes inteligentes implementados através de sensores capazes de absorver informações do ambiente e reagir de maneira autônoma através das informações obtidas, de modo a atingir um objetivo. Além disso, os agentes podem se comunicar entre si. Um agente pode ser definido como um *software* que possui um objetivo a ser atingido em relação ao ambiente em que está atuando (Wooldridge 2009).

Com o intuito de estudar problemas e soluções para um *Smart Parking* do tipo ABGS, criou-se o projeto *Multi-Agent Parking System* (MAPS) (Castro *et al*. 2016a) e (Castro *et al.* 2016b). Neste projeto existem diferentes linhas de estudo, como implementação de agentes, organização social, negociação, coordenação, dentre outros. Ao passo que o trabalho aqui apresentado tem como foco principal facilitar a compreensão do comportamento de sistemas multi-agentes por meio da integração de ferramentas de simulação que oferecem recursos gráficos para análise visual de tal comportamento. Neste projeto é apresentado a simulação de estacionamentos controlados (ambientes que possuem controle de acesso e não estacionamentos públicos) por meio do simulador *Simulation of Urban MObility* (SUMO).

Para que fique mais claro o contexto de aplicação do SUMO, faz-se necessário explicar em detalhes o funcionamento do projeto MAPS. O modelo de alocação de vagas implementado pelo MAPS trabalha com dois tipos de agentes: o agente motorista (*driver*) e um agente gerente (*manager*). O primeiro é o agente o qual interage no ambiente, ou seja, os motoristas que estão em busca de uma vaga para estacionar. O segundo é o agente

centralizador do sistema multiagente (SMA), responsável por gerenciar as vagas do estacionamento e alocá-las aos agentes motoristas. As vagas (*spots*) são consideradas os recursos do SMA gerenciados pelo *manager* e atribuídos ao *driver*. Quando todas as vagas do estacionamento estão ocupadas, os *drivers* que chegarem ao estacionamento são direcionados para uma fila de espera. Ao liberar uma nova vaga, um *driver* é selecionado conforme o respectivo grau de confiança ou então quando um tempo limite de permanência na fila é atingido, por exemplo, 60 segundos, sendo que este valor pode ser configurado (Castro *et al.* 2016b).

O *manager* é o responsável por manter a comunicação entre os outros agentes que estão no ambiente e também alocar as vagas para os *drivers*. Cada *driver* possui um grau de confiança (*trust degree*), sendo que esse grau é determinado e calculado pelo agente *manager*. Especificamente, o grau de confiança de cada agende *driver* (D) é atribuído conforme a seguinte fórmula (Castro *et al.* 2016a):

trustDegree(D) ::= trustDegree(D) + a + t + u

onde,

- **attendance (a):** valor referente à frequência de uso do estacionamento pelo *driver*;
- **time of usage (t):** valor referente ao tempo gasto pelo *driver* no estacionamento;
- **usage (u):** valor referente ao comportamento do *driver* dentro do estacionamento. O comportamento é medido através de regras sociais implementadas no nível de organização do projeto. Por exemplo, se o *manager* definir uma vaga para o *driver* e o mesmo não a ocupar, o *driver* será penalizado.

A alocação de vagas é feita através do grau de confiança dos *drivers*. O *driver* que irá receber a vaga é aquele que possui a maior grau de confiança entre todos os que estão na fila de espera. No momento em que o *driver* deixa a vaga, o sistema atribui um dado valor ao agente, o qual poderá incrementar ou decrementar o respectivo grau de confiança. Na figura 1 é possível observar um esquema geral do sistema, onde o *driver* D1 obtém a vaga V6, visto que possui um grau de confiança maior que o *driver* D2.

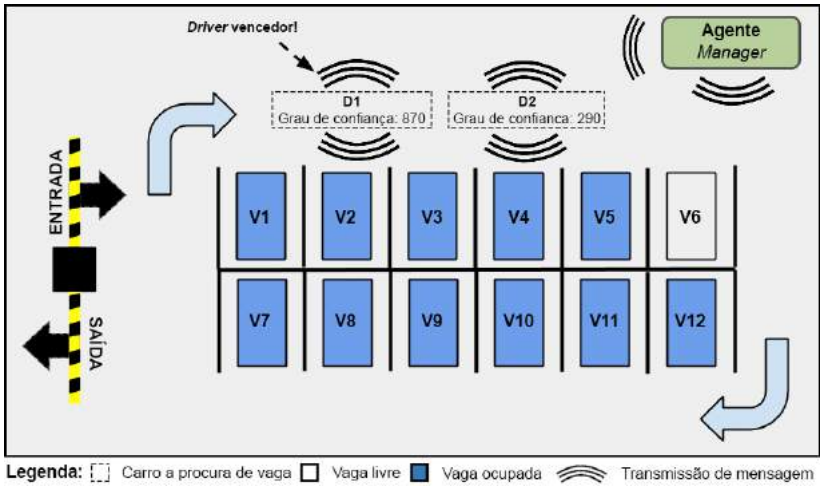

Figura 1 – Exemplo de funcionamento do sistema implementado pelo MAPS

O SUMO é um simulador de tráfego urbano de código aberto, que permite a modelagem de diversos tipos de ambientes e a simulação do comportamento desses ambientes com fluxo de tráfegos variados, aproximando-se ao máximo do mundo real (Behrisch et al. 2011). A integração entre o SUMO e o MAPS busca justamente preencher uma lacuna do projeto MAPS em relação ao uso de simuladores de trânsito que possam ajudar a obter resultados melhores e mais precisos. O uso de simuladores permite visualizar graficamente o funcionamento do ambiente e contribui para o teste de novos algoritmos de alocação de vagas implementados no projeto.

Por sua vez, o projeto MAPS é implementado utilizando o *framework* para desenvolvimento de sistemas multiagentes JaCaMo[1], o qual é dividido em três níveis: nível dos agentes (Jason), nível do ambiente (Cartago) e nível de organização (Moise) (JaCaMo 2011). A integração entre o SUMO e o *framework* JaCaMo dá-se por meio de um artefato, desenvolvido no nível de ambiente do projeto MAPS, e do *Middleware* MAPS-SUMO[2], ferramenta intermediária dessa conexão, desenvolvida por este projeto, a qual recebe protocolos do MAPS e encaminha para a simulação. O objetivo principal deste trabalho é demonstrar a possibilidade de integração entre o JaCaMo e o SUMO, bem como a viabilidade de uma representação gráfica do uso do estacionamento por meio de uma ferramenta de simulação. O projeto MAPS é utilizado como um exemplo de uso do *framework* JaCaMo para demonstrar a aplicação prática da integração entre as ferramentas e, a partir disso, acredita-se que será possível estabelecer uma integração genérica entre o SUMO e o JaCaMo.

O restante deste artigo descreve o seguinte: a Seção 2 apresenta os trabalhos relacionados; a Seção 3 discorre sobre a ferramenta SUMO; a Seção 4 exibe o funcionamento do *Middleware* MAPS-SUMO; a Seção 5 demonstra os resultados obtidos e, por fim, a Seção 6 apresenta as conclusões deste artigo.

2. Trabalhos Relacionados

Esta Seção tem como objetivo apresentar trabalhos relacionados que englobam as áreas de mobilidade urbana, trânsito, *Smart Parking* e SUMO.

Especificamente, aqui destacam-se três diferentes trabalhos que demonstram a aplicação da ferramenta SUMO em problemas de trânsito, sendo que dois dos três trabalhos ainda apresentam o uso de agentes inteligentes. Inicialmente, apresenta-se o trabalho de Krajzewicz *et al*. (2012), onde os autores criam um modelo denominado CityMobil, no SUMO, o qual simula um ambiente de estacionamento de transporte público, onde ônibus circulam em uma rota e param em alguns locais para o embarque de passageiros. Carros são gerados em fluxos e percorrem a rota até estacionarem. Porém os veículos sempre seguem uma mesma ordem na hora da escolha de vagas e não há um controle de vagas livres e ocupadas. Através da biblioteca TraCI[3], cria-se uma interface para controlar carros chegando, pessoas entrando na fila de espera e pessoas embarcando nos ônibus.

[1] Ao longo do texto o termo JaCaMo é utilizado para referenciar o framework usado, porém neste trabalho utilizou-se somente as linguagens Jason e Cartago.
[2] MAPS-SUMO é o termo utilizado para definir o Middleware de conexão entre o MAPS e o SUMO.
[3] Biblioteca TraCI4J (https://github.com/ergueli/TraCI4J).

Outro trabalho que igualmente faz uso do SUMO é apresentado por Baines e Padget (2014), os quais descrevem o desenvolvimento de um *framework* para analisar o comportamento de veículos inteligentes. Para a simulação foi utilizada a ferramenta SUMO e para o controle de agentes foi utilizado a linguagem Jason. Em um dos cenários de teste, o Jason adiciona veículos (agentes) ao SUMO especificando algumas características e faz verificações para detectar possíveis colisões, e então reduzir a velocidade do veículo ou trocar de faixa. Em um segundo cenário, veículos são inseridos no SUMO e detectam semáforos e seu estado (aberto, fechado). O agente recebe uma ordem para reduzir a velocidade para um determinado valor e por um determinado período de tempo para chegar ao semáforo aberto, e então o controle do agente é devolvido ao SUMO. O trabalho de Baines e Padget (2014) aborda uma maneira de trabalhar com o SUMO utilizando os veículos como agentes controlados pela linguagem Jason, interligando as duas ferramentas através do desenvolvimento de um *framework* distribuído. Este *framework* inclui também ferramentas de simulação 3D, ambiente *Android* e ferramentas de *debug*. Porém o estudo foi feito em relação a simulação de trânsito em geral, sendo que o foco dos autores não considerava o gerenciamento de estacionamentos, como é o caso do projeto MAPS.

Por fim, ainda destaca-se o trabalho proposto por Batista Júnior e Coutinho (2013), onde os autores descrevem um sistema multiagente o qual atua em vias arteriais, controladas por semáforos, com o objetivo de tornar os cruzamentos mais eficientes. Cada intersecção é controlada por um agente. Os agentes são controlados através da linguagem de programação de agentes Jason e simulados na ferramenta SUMO. O artigo também demonstra o modelo de integração entra as duas ferramentas, que é feito através da API (*Application Programming Interface*) XTRACI, desenvolvida em java e baseada na biblioteca TraCI, a mesma utilizada pelo modelo CityMobil.

Os trabalhos previamente citados evidenciam o uso de soluções tecnológicas para *Smart Parking*, bem como o uso do simulador SUMO em conjunto com agentes inteligentes para simulações de trânsito. Contudo, pelo que se conhece da literatura atual, não foram encontrados trabalhos que demonstram a interligação entre o simulador SUMO e o *framework* JaCaMo, apenas a interligação com a linguagem de programação Jason. Além disso, os trabalhos que mostram o uso do Jason com o SUMO têm foco na simulação de fluxo de trânsito. Ao passo que o trabalho aqui apresentado tem foco no gerenciamento de vagas de estacionamento.

3. Ferramenta SUMO

O SUMO é uma ferramenta de simulação desenvolvida pelo Instituto de Transporte da Alemanha com o intuito de aprimorar os resultados finais de um projeto e suas análises, além de facilitar os testes de novos algoritmos que são desenvolvidos nessa área (Krajzewicz et al. 2006).

O simulador SUMO oferece diversas funcionalidades ao usuário, como simulação microscópica; controle de tempo de semáforos; importação de ambientes reais através da ferramenta NETCONVERT em conjunto com OpenStreetMaps; o TraCI, que permite

obter valores da simulação e manipulá-los em tempo real, dentre muitas outras possibilidades.

Para este trabalho, foi feito um estudo de caso utilizando o modelo criado pela ferramenta CityMobil, a qual simula um ambiente de estacionamento semelhante a um aeroporto, com algumas adaptações (Behrisch et al. 2011). Faz-se necessário destacar que o modelo do CityMobil serve apenas para fins de testes da interligação do SUMO com o projeto MAPS. O modelo, que pode ser observado na figura 2, conta com 160 vagas divididas em 9 setores: A, B, C, D, E, F, G, H e I.

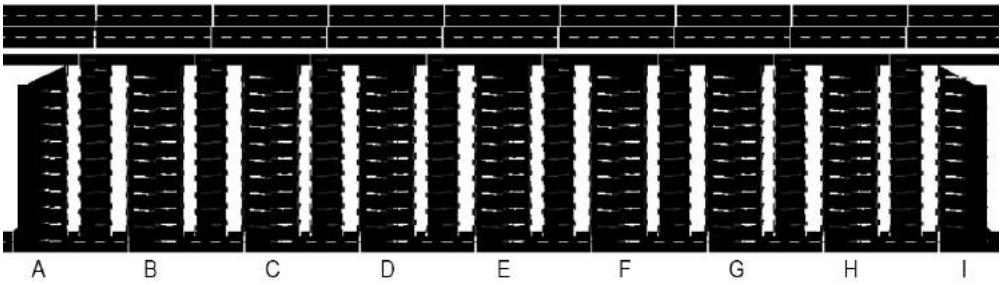

Figura 2 – Visão do modelo de estacionamento usado no MAPS-SUMO

O SUMO é responsável apenas pela modelagem do ambiente e pela definição das vagas e e rotas de cada vaga. A administração do ambiente, como controle de entrada e saída de veículos, é feita diretamente no projeto MAPS, portanto, o SUMO é responsável apenas por exibir os resultados de acordo com as instruções recebidas do *Middleware*.

4. *Middleware* MAPS-SUMO

A integração entre o projeto MAPS e a ferramenta SUMO dá-se através da criação de uma arquitetura cliente-servidor. Para isso, foi construído o *Middleware* MAPS-SUMO, desenvolvido na linguagem de programação Java e que trabalha como intermediário entre o MAPS e o SUMO. O MAPS trabalha como cliente e o M*iddleware* como servidor. No *Middleware* está integrada a biblioteca TraCI4J, baseado no TraCI, responsável pela comunicação com o SUMO.

Portanto, o MAPS-SUMO funciona como servidor, aguardando mensagens enviadas pelo projeto MAPS, e também como cliente, tratando as mensagens recebidas e encaminhando para o simulador SUMO. Na figura 3 é possível observar a visão geral da comunicação entre as aplicações.

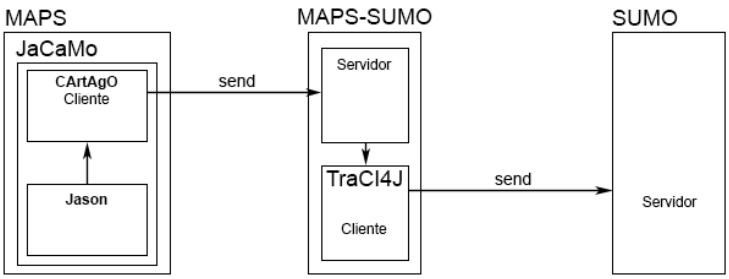

Figura 3 – Visão geral da integração entre o projeto MAPS e o SUMO

A comunicação do projeto MAPS com o MAPS-SUMO é feita por meio de um artefato do *framework* JaCaMo, no nível de ambiente Cartago e, devido ao fato do mesmo ser programado em Java, isto facilita a comunicação entre os dois. Um artefato é definido como uma entidade não-autônoma dentro de um ambiente, a qual é invocada somente pelos agentes. O artefato é acessado por meio do agente *manager* criado em Jason.

Esse artefato oferece as três principais operações necessárias para estabelecer a comunicação com a simulação no momento da execução: (i) alocar uma vaga (*allocateSpotSumo*), (ii) deixar o estacionamento (*leaveSpotSumo*) e (iii) inserir motorista na fila de espera (*insertDriverQueueSumo*). Essas operações são executadas pelo agente *manager*, o qual controla todas as requisições feitas pelos agentes *drivers*.

No momento em que o agente *manager* executa uma das três operações citadas anteriormente, o mesmo dispara um chamado para o artefato criado, enviando como parâmetro a identificação do agente participante da operação, e, se necessário, a vaga a qual o agente irá ocupar ou desocupar. O artefato abre uma comunicação com o servidor MAPS-SUMO e encaminha um protocolo. Este protocolo foi desenvolvido com o intuito de padronizar e simplificar a comunicação entre as ferramentas. O protocolo é definido pela seguinte tupla:

Protocolo ::= <idDriver, nomeDriver, tipoProtocolo, idSpot, trustDriver>

onde,

- **idDriver**: é um campo identificador do agente *driver*, sempre preenchido com a letra D, indicando que, em seguida, aparece o nome do agente *driver*;
- **nomeDriver***:* identificação do agente *driver*;
- **tipoProtocolo***:* tipo do protocolo, que está relacionado com a operação em que ele foi gerado: PS (*Parking Spot*) para alocação de vaga, LS (*Leaving Spot*) para agentes deixando o estacionamento e QU (*Queue*) para inserir um motorista na fila de espera;
- **idSpot***:* identificação da vaga a ser ocupada ou desocupada. Caso o protocolo seja do tipo QU, este campo é ignorado;
- **trustDriver***:* sempre precedido das letras TR, as quais indicam que será definido um grau de confiança para este agente. Necessário apenas para protocolos referentes à alocação de vaga.

Na figura 4 é possível observar um exemplo de um protocolo de comunicação pronto, o qual representa a alocação da vaga *spotB10* para o agente *driver07*.

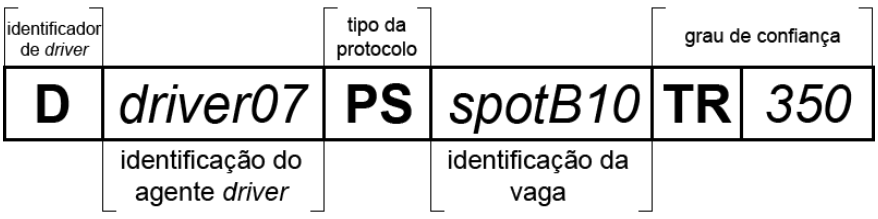

Figura 4- Exemplo de protocolo de comunicação entre MAPS e *Middleware*

Para efetivar o envio de mensagens dos agentes para o MAPS-SUMO foi desenvolvido um artefato em Cartago chamado *Simulation*, responsável por criar os protocolos específicos para cada operação e encaminhá-los para a classe de comunicação *TCPConnection*. A comunicação entre o agente *manager*, o artefato e a classe de comunicação é demonstrada por meio do diagrama 1.

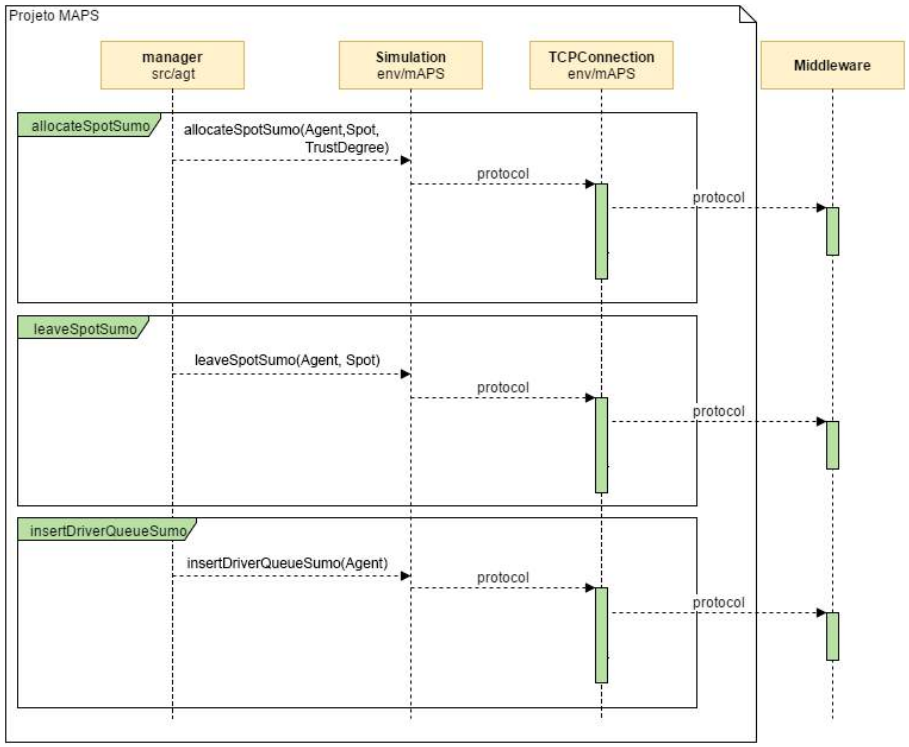

Diagrama 1 – Comunicação entre o agente manager, o artefato *Simulation* e a classe de comunicação

A criação do artefato é feita pelo agente *manager* e com isso o agente tem acesso às operações disponíveis no mesmo e podem ser chamadas a qualquer momento. O artefato é responsável por definir o protocolo de comunicação de acordo com os parâmetros enviados pelo agente e encaminhá-lo ao MAPS-SUMO, responsável por transformar o protocolo em mensagens e transmití-las ao SUMO. Na operação *allocateSpotSumo* (código 1) é possível notar a criação do protocolo e sua transmissão para o MAPS-SUMO.

```
@OPERATION
public void allocateSpotSumo(Object idDriver, Object idSpot, Object trustDegree) {
        String protocol = "D" + idDriver + "PS" + idSpot + "TR" + trustDegree;
        conn.connect(protocol);
}
```

Código 1 – Operação allocateSpotSumo

O MAPS-SUMO é o elemento chave da integração entre o projeto MAPS e a ferramenta SUMO. Atua como servidor, recebendo protocolos de comunicação do projeto MAPS relacionados às decisões tomadas pelo agente *manager*. O próprio MAPS-SUMO

decodifica esses protocolos e os transforma em mensagens, que serão transmitidas para o SUMO através da biblioteca TraCI4J (diagrama 2).

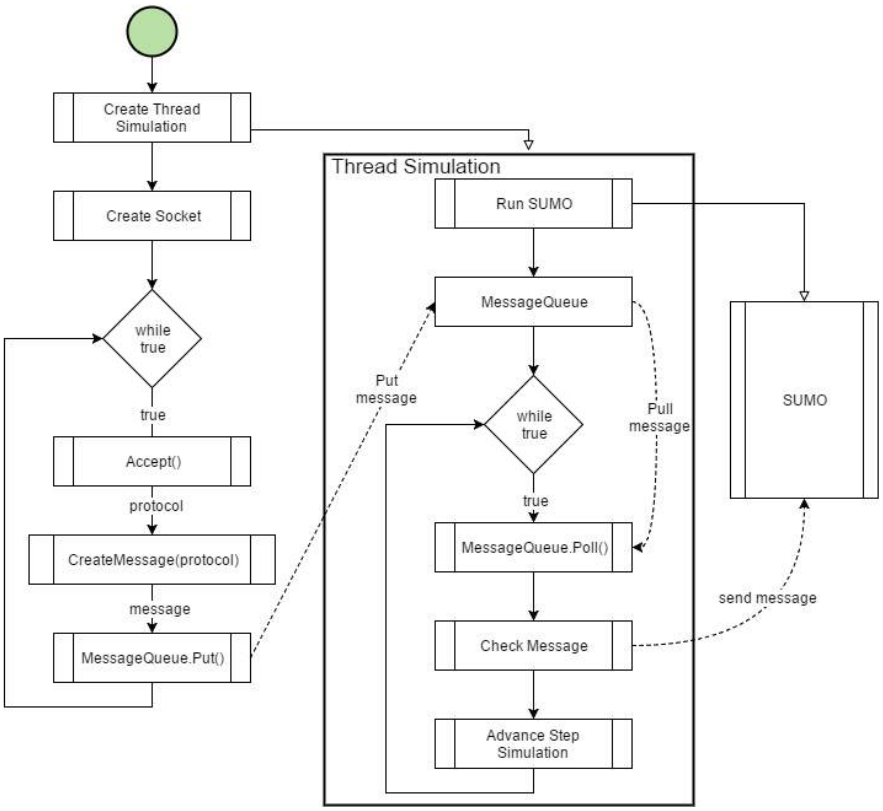

Diagrama 2 - Visão geral do funcionamento do MAPS-SUMO

O diagrama 2 apresenta uma visão geral do funcionamento do MAPS-SUMO. Primeiramente é criada uma *thread*, a qual executa a simulação gráfica, cria uma lista de mensagens e aguarda novas mensagens a serem enviadas à simulação. Todos os procedimentos de comunicação direta com o SUMO são controlados pela biblioteca Traci4J. Em paralelo, o MAPS-SUMO cria o servidor responsável por receber os protocolos enviados pelo MAPS, transformá-los em mensagens e encaminhá-las para a *thread*.

A mensagem obtida por meio do protocolo é encaminhada para a *thread* onde situa-se a biblioteca Traci4J, responsável pela comunicação com o SUMO. A *thread* segue basicamente quatro passos: (i) receber uma mensagem do MAPS-SUMO; (ii) verificar o tipo da mensagem; (iii) encaminhá-la para simulação; e (iv) avançar a simulação.

Por fim, a ferramenta SUMO é responsável por receber comandos encaminhados pelo MAPS-SUMO e demonstrar de maneira gráfica o funcionamento do ambiente de estacionamento em tempo real. O MAPS-SUMO é responsável por encaminhar as mensagens ao SUMO, através da biblioteca Traci4J. O próprio SUMO trabalha como servidor, rodando paralelamente ao MAPS-SUMO e aguardando mensagens para exibir na simulação.

5. Resultados

A partir do desenvolvimento do MAPS-SUMO apresentado na Seção 4, foi possível analisar o comportamento do projeto MAPS de maneira gráfica. O quadro 1 detalha as cores dos veículos conforme o respectivo valor do grau de confiança do *driver*.

Quadro 1 – Graus de confiança dos agentes e suas respectivas cores na simulação

Grau de confiança	Cor
0 a 100	Vermelho
101 a 200	Verde
201 a 300	Azul
301 a 400	Amarelo
Até 999	Branco

A partir deste quadro, pode-se observar na figura 5 a aplicação de cores na simulação de acordo com o grau de confiança dos agentes *drivers*, onde 10% dos *drivers* são representados pela cor vermelha, 34% pela cor verde, 20% pela cor azul, 26% pela cor amarela e 10% pela cor branca. Nota-se que, quando se utilizam cores, é mais simples visualizar o resultado da simulação e compreender como ocorre a ocupação do estacionamento em relação aos valores de confiança dos *drivers.*

Figura 5 – Cores dos veículos baseado em seu grau de confiança

Com a conexão entre o projeto MAPS e a ferramenta da simulação SUMO, a análise do comportamento do ambiente se torna mais prática, onde é possível observar visualmente a quantidade de vagas livres e ocupadas, além da possibilidade de acompanhar o trânsito dos veículos dentro do ambiente, como mostra a figura 6. Nesta figura nota-se que há 99 vagas ocupadas de um total de 160, o que corresponde à 61,87% de ocupação do ambiente, onde 16,88% são veículos verdes, 13,12% azuis, 11,87% amarelos, 13,75% brancos e 6,25% vermelhos. Ademais, percebe-se que os últimos três setores estão desocupados.

Figura 6 - Alocação de vagas no estacionamento

Para resultados que discutam a relevância do grau de confiança na alocação de vagas, sugere-se a leitura de Castro *et al.* 2016b, visto que as simulações realizadas aqui foram excutadas para evidenciar o correto funcionamento do *Middleware*, bem como destacar o uso do simulador gráfico com cores no realce dos diferentes graus de confiança.

Para efetivar a comunicação entre o projeto MAPS e a ferramenta SUMO, o *middleware* tornou-se responsável por receber protocolos do MAPS, transformá-los em mensagens e transmití-las para o SUMO. Apesar desse procedimento de comunicação intermediário, concluiu-se que não houve perdas de protocolos e mensagens, uma vez que os mesmos foram gravados em três momentos: (i) criação/transmissão do protocolo pelo MAPS; (ii) recepção do protocolo e transmissão da mensagem pelo MAPS-SUMO; e (iii) recepção da mensagem pelo SUMO.

6. Conclusão

O objetivo principal deste trabalho é apresentar a integração entre o *framework* JaCaMo e a ferramenta de simulação SUMO. O projeto MAPS foi utilizado como um exemplo de aplicação prática do *framework* JaCaMo para demonstrar a viabilidade da integração entre as ferramentas. Por meio do desenvolvimento deste trabalho, os seguintes resultados foram obtidos: (i) representação gráfica da simulação do uso de um estacionamento, o que viabiliza a implantação de novos algoritmos e testes ao projeto MAPS; (ii) criação de um protocolo de comunicação entre JaCaMo e SUMO. Salienta-se que (até onde se tem conhecimento) este trabalho é pioneiro ao demonstrar a integração do *framework* JaCaMo com o simulador SUMO.

A integração entre as ferramentas foi possível através da criação de um protocolo de comunicação, responsável por transmitir informações entre os agentes e a simulação gráfica. Os protocolos foram criados em um artefato situado no ambiente Cartago, e são chamados pelos agentes, programados em Jason. Além de montar os protocolos, o artefato também foi responsável por transmitir as mensagens para o MAPS-SUMO, ferramenta desenvolvida por este projeto e responsável por intermediar a comunicação entre o *framework* JaCaMo e o SUMO.

A criação do protocolo de comunicação pode também viabilizar a definição de uma futura comunicação genérica entre o JaCaMo e o SUMO, onde pretende-se utilizar o Cartago apenas para efetivar essa comunicação. As classes *allocateSpotSumo*, *leaveSpotSumo* e *insertDriverQueueSumo* seriam removidas por serem referentes ao projeto MAPS, e o protocolo seria um possível parâmetro recebido pela classe de comunicação definida no Cartago. Ainda como outro possível trabalho futuro, almeja-se implantar outros modelos de estacionamentos, além do CityMobil utilizado neste projeto.

Referências

Baines, V. e Padget, J. “A situational awareness approach to intelligent vehicle agents”, SUMO2014 – Modeling Mobitlity with Open Data. Berlim - Alemanha, v. 24, p. 1-17, maio 2014.

Batista Júnior, A. A. E Coutinho, L. R. “Incorporating explicit coordination mechanisms by agents to obtain green waves”, Advanced Methods and Technologies for Agent and Multi-Agent Systems, v. 252, p. 137-145, 2013.

Batty, M, Axhausen, K., *et al*. “Smart Cities of the Futures”, UCL Working Papers Series, Londres - Inglaterra, v. 188, out. 2012.

Behrisch, M., Bieker, L., Erdmann, J., E Krajzewicz, D. “SUMO – Simulation of Urban Mobility”, SIMUL 2011 – The Third International Conference on Advances in System Simulation, Barcelona, Espanha, 2011.

Castro, L. F. S., Alves, G. V., Borges, A. P. “A proposal of an architecture for a Smart Parking based on intelligent agents and embedded systems”, In: Anais do WPCCG (I Workshop de Pesquisa em Computação dos Campos Gerais) 2016, v. 1, p. 59-61, ISSN: 2526-1371, Editora: UTFPR, 2016 a.

Castro, L. F. S., Alves, G. V., Borges, A. P. “Utilização de grau de confiança entre agentes para alocação de vagas em um Smart Parking”, In: Anais do WESAAC 2016 (Workshop – Escola de Sistemas de Agentes, seus Ambientes e Aplicações), 2016 b.

Denatran. “Departamento Nacional de Trânsito”, http://www.denatran.gov.br/frota2016.htm, Dezembro.

Deutsche Telekom. “Deutsche Telekom and Pisa start Smart City project”, Mobile World Congress, Barcelona, Espanha, 2014.

JaCaMo. “The JaCaMo Project. Multi-Agent Programming Framework”, http://jacamo.sourceforge.net, Dezembro.

Koster, A., Koch, F. E Bazzan, A. “Incentivising Crowdsourced Parking Solutions”, Citizen in Sensor Networks, CitSens, Barcelona, Espanha, v. 8313, set. 2014.

Krajzewicz, D., Bonert, M. E WagneR, P. “The open source traffic simulation package SUMO”, RoboCup 2006 Infraestructure Simulation Competition. Berlim, Alemanha, 2006.

Krajzewicz, D., Erdmann, J., Behrisch, M. E Bieker, L. “Recent development and applications of SUMO – Simulation of Urban MObility”, International Journal On Advances in Systems and Measurements, Berlim, Alemanha, v. 5, n. 3 & 4, p. 128-138, 2012.

Revathi, G. E Dhulipala, V. “Smart parking Systems and Sensors: A Survey”, Computing, Communication and Applications, Dindigul, India, p. 1-5, fev. 2012.

Wooldridge, M. “An Introduction to MultiAgent Systems”, 2nd. Ed. [S.1.]: Wiley Publishing, 2009.