Integrando Sistemas Multi-Agentes Embarcados, Simulação Urbana e Aplicações de IoT *

Lucas Fernando Souza de Castro[1], **Fabian Cesar P. B. Manoel**[2]
Vinícius Souza de Jesus [2], **Carlos Eduardo Pantoja**[2]
André Pinz Borges[3], **Gleifer Vaz Alves**[3]

[1]Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Campinas – SP – Brasil

[2]Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET/RJ)
Rio de Janeiro – RJ – Brasil

[3]Programa de Pós-Graduação em Ciência da Computação
Universidade Tecnológica Federal do Paraná (UTFPR)
Ponta Grossa, PR – Brasil

lucas.castro@ic.unicamp.br, {souza.vdj, fabiancpbm}@gmail.com

pantoja@cefet-rj.br, {apborges, gleifer}@utfpr.edu.br

Abstract. *The development of smart city systems connected to the Internet of Things (IoT) has been the goal of several multi-agent system research works. However, few works show how to deploy and make the connection among the employed systems. This paper proposes an approach towards the integration of a MAS using JaCaMo framework along with the Urban Simulation Tool (SUMO), and some IoT applications. The integration presented here is applied in a Smart Parking scenario with real features, where is shown the integration and the connection throughout all layers, from agent level to artifacts, including real environment and simulation, as well as IoT applications. In future works, we intend to establish a methodology which shows how to properly integrate these different applications no matter the applied scenario nor the used tools.*

Resumo. *O desenvolvimento de sistemas para cidades inteligentes conectados com Internet of Things (IoT) tem sido o foco de muitas pesquisas no âmbito de Sistemas Multi-Agentes (SMA). Contudo, poucos trabalhos mostram como realizar a implantação e conexão entre os softwares utilizados nestas diferentes áreas. Assim, este trabalho propõe uma abordagem de integração de um SMA, usando o framework JaCaMo com a ferramenta de Simulação Urbana (SUMO) e aplicações de IoT. A integração apresentada aqui é aplicada em um cenário de Estacionamento Inteligente com característica realista, onde a integração e conexão de todos componentes envolvidos é descrita desde o nível dos agentes e artefatos, passando pelo nível do ambiente real (físico), pelo nível da simulação*

urbana, chegando as aplicações de IoT. Em trabalhos futuros, pretende-se elaborar uma metodologia que mostre como integrar essas diferentes aplicações independentemente dos cenários e das ferramentas utilizadas.

1. Introdução

A Cisco em um de seus relatórios aponta uma estimativa de aproximadamente 50 bilhões de dispositivos para a *Internet of Things* (IoT) em funcionamento a partir de 2022 [Evans 2011]. Além disso, há uma crescente demanda de sistemas computacionais que proporcionem funcionalidades e conectividade para esses dispositivos. O desenvolvimento de tais sistemas deve atender a dinamicidade dos cenários onde os dispositivos para IoT estão localizados, como por exemplo o caso de uma cidade inteligente.

Uma cidade inteligente pode ser definida como uma cidade que provê tecnologias integradas ao seus cidadãos para que tenham qualidade de vida [Albino et al. 2015]. Essa qualidade fornecida à população pode ser diversificada em várias categorias, sendo: governabilidade, segurança, economia, meio ambiente e mobilidade. Como demonstrado em [Neirotti et al. 2014], a questão da mobilidade tem sido uma área de atenção de pesquisa e investimento industrial em cidades inteligentes. Assim, é necessário diante de tais demandas o desenvolvimento de sistemas capazes de fornecer suporte à mobilidade dos cidadãos.

Os Sistemas Multi-Agentes (SMA) são candidatos a fornecer o suporte necessário à mobilidade dos cidadãos, pois são sistemas compostos por múltiplos agentes autônomos, independentes e pro-ativos com raciocínio cognitivo e capacidade de tomada de decisão com habilidade de comunicação entre si, a fim de atingir um objetivo comum ou conflitante [Wooldridge 2000]. Tais características permitem que problemas relacionados à mobilidade em cidades inteligentes possam ser tratados de forma pro-ativa sem interferência ou até conhecimento dos cidadãos, porque agentes inteligentes podem assumir esse papel em benefício do usuário e negociar com outros agentes para tomar as decisões mais pertinentes em dado momento para qualquer serviço. Como em uma cidade inteligente podem existir inúmeros dispositivos capazes de trocar informações, a IoT surge como uma tecnologia essencial na implantação de tais soluções, já que, por definição, IoT é um conjunto de dispositivos interconectados pela Internet. Visto que SMA podem ser explorados tanto de forma simulada (virtual), quanto de forma física, é possível prever dispositivos embarcados com SMA aplicados em aplicações de IoT.

Ao observar a demanda tecnológica requerida pela mobilidade urbana, observa-se a necessidade de um sistema adequado para lidar com situações de inteligência e organização em um ambiente, e sobretudo com suporte da IoT para escalar a resolução do problema. Sendo assim, o objetivo deste trabalho é descrever uma abordagem de integração de diferentes aplicações, envolvendo SMA, Simulação Urbana, Sistemas Embarcados e a Camada IoT de forma que seja possível criar soluções que utilizem tais tecnologias em conjunto para resolver questões de mobilidade, heterogeneidade e baixo acoplamento entre as tecnologias. Consequentemente, essa integração visa prover tanto uma solução capaz de ser aplicada em diferentes cenários e aplicações (heterogeneidade), quanto uma solução que proporcione a possibilidade de utilizar somente uma parte - subconjunta - da integração como uma solução (baixo acoplamento entre as tecnologias).

Para implantação do trabalho utiliza-se o *framework* JaCaMo

[Boissier et al. 2013], uma composição de outros três *frameworks*, Jason, CArtAgO e Moise. Jason [Bordini et al. 2007] interpreta uma linguagem orientada a agentes denominada *AgentSpeak* em Java para a programação de agentes BDI (*Belief-Desire-Intention*). Já o CArtAgO é baseado no modelo *Agents & Artifacts (A&A)* o qual permite o desenvolvimento da camada ambientes do SMA e provê a integração entre os agentes e artefatos de um SMA [Ricci et al. 2006]. Por fim, o Moise implementa um modelo organizacional para SMA que tem fundamentação nos conceitos de agrupamento, comportamento e objetivos [Boissier et al. 2013].

O restante deste artigo está organizado da seguinte maneira. Seção 2 apresenta a abordagem para integração dos sistemas. Na Seção 3 é descrita a integração entre os agentes e artefatos do SMA com a simulação. Após, a Seção 4 detalha a integração de agentes e artefatos com o meio físico. Enquanto, na Seção 5, a integração de agentes com a IoT é descrita. Por fim, a Seção 6 discute as considerações finais.

2. Abordagem para Integração de Sistemas

Nesta seção é apresentada a abordagem para integração dos diferentes Sistemas: SMA, Simulação Urbana, Sistemas Embarcados e a camada de IoT. A abordagem utilizada descreve como desenvolver um SMA embarcado para aplicações de IoT utilizando o framework JaCaMo e integrando diversas ferramentas de forma a facilitar a aplicação e visualização de tais sistemas, inclusive em diferentes domínios de aplicação. A abordagem empregada aqui permite a separação da estrutura de um sistema em níveis onde as responsabilidades do software, hardware e IoT são programadas separadamente para facilitar a integração dependendo da solução que está sendo criada e de quais níveis serão utilizados. A proposta de integração de sistema é composta por quatro níveis bem definidos: Agentes, Ambiente, IoT e nível da Simulação, conforme ilustrado na Fig. 1.

A camada de nível do *agente* é responsável pela criação dos agentes do SMA com uso da linguagem Jason. Os agentes possuem duas extensões específicas: uma para o interfaceamento de hardware usando o ARGO [Pantoja et al. 2016], e uma outra para comunicação e transporte de agentes usando a IoT. Esta camada de nível relaciona-se com o nível de ambiente através dos artefatos disponibilizados pelo CArtAgO [Ricci et al. 2006], com o nível de simulação para que seja possível simular os agentes utilizando a ferramenta SUMO [Krajzewicz et al. 2002], com o nível de ambiente real diretamente através do agente estendido ARGO, e por fim, com o nível IoT para escalar as aplicações com agentes.

O nível do *ambiente* é dividido entre os artefatos que podem ser criados pelo CArtAgO e pelo meio físico, onde as plataformas de hardware com sensores e atuadores podem interagir com o mundo físico. Assim é possível criar os *Artefatos Simulados*, aqueles que mantém informações apenas em nível de software, e os Artefatos Físicos, que realizam a interface de Hardware por meio da utilização de uma Interface Serial. Ambos artefatos podem co-existir em uma solução desenvolvida usando a abordagem proposta, uma vez que as tecnologias utilizadas em cada nível são independentes. Os Artefatos Físicos podem também se relacionar com o nível de IoT via de conexões MQTT com o *KonkerLabs* e o *Node-Red Application*.

No nível da IoT, é utilizado o *ContextNet*, um *middleware* para IoT que tem capacidade para lidar com diversas características de sistemas distribuídos como conecti-

vidade, escalabilidade e comunicabilidade, para comunicação entre agentes. Os agentes comunicadores da abordagem proposta são capazes de conectar-se ao *ContextNet* para trocar mensagens ou realizar o transporte de agentes de um SMA embarcado em um dispositivo para um outro SMA de um dispositivo diferente.

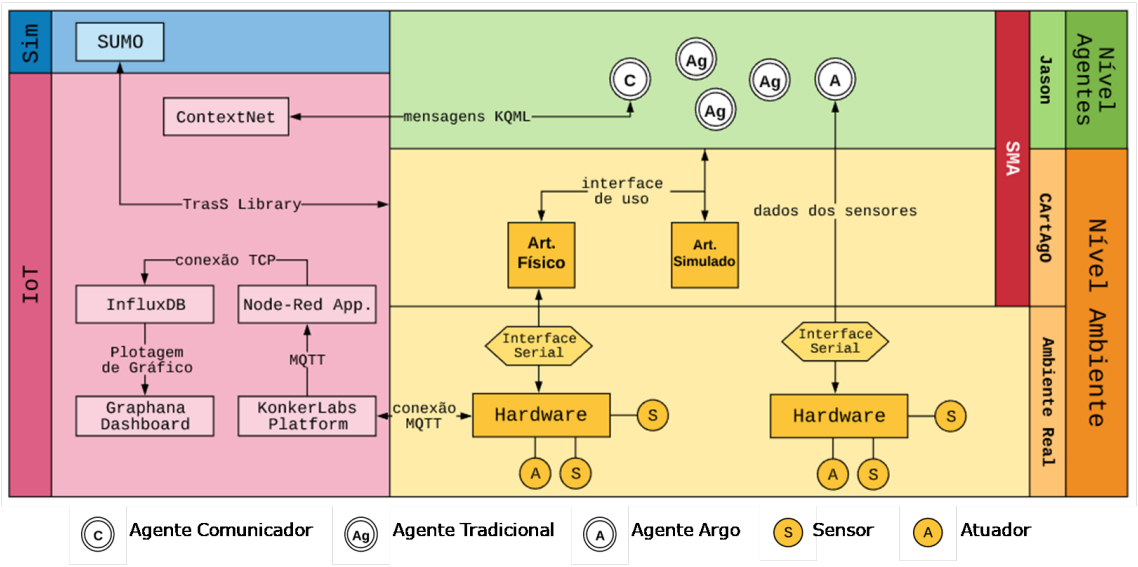

Figura 1. Abordagem para Integração de Sistemas: visão geral

Dessa forma, é possível criar dispositivos utilizando a abordagem de agentes para atuar fisicamente em um Sistema Ciber-Físico e aplicações de IoT. Pode-se definir um Dispositivo como um componente composto de um SMA embarcado utilizando o Jason e o CArtAgO, capaz de interfacear sensores e controladores conectados a hardwares microcontrolados (ATMEGA, PIC, Arduino, etc.) e de se conectar a uma rede IoT para troca de informações.

3. Integração: Agentes, Ambiente (Artefatos) e Simulação

A utilização do *framework* JaCaMo, além de proporcionar três camadas de desenvolvimento para os SMAs: agentes, artefatos e social, possui também a capacidade de interligação com diferentes ferramentas, por exemplo: simulação, sistemas embarcados e IoT. Tal capacidade é derivada da versatilidade do *framework* em prover por meio da utilização de artefatos de ambiente do CArtAgO. Essa seção descreve como é realizada a conexão do JaCaMo e SUMO.

Para ilustrar essa integração é utilizado um SMA desenvolvido no JaCaMo para alocação e negociação de vagas em Estacionamento Inteligentes. Na simulação com o SUMO, utiliza-se uma rede[1] da UTFPR - Campus Ponta Grossa com enfoque no estacionamento de visitantes conforme ilustrado na Figura 2.

A integração do SMA com o SUMO foi divida em duas camadas: agentes e artefatos. A camada dos agentes compreende a programação dos agentes na linguagem Jason e suas interações sociais. Enquanto a segunda camada apresenta os artefatos desenvolvidos em CArtAgO e suas interações com os agentes. Ambas as camadas são descritas em detalhes na seção seguinte.

[1]O termo rede no SUMO (ou *network*) é utilizado para denotar um mapa a ser empregado na simulação.

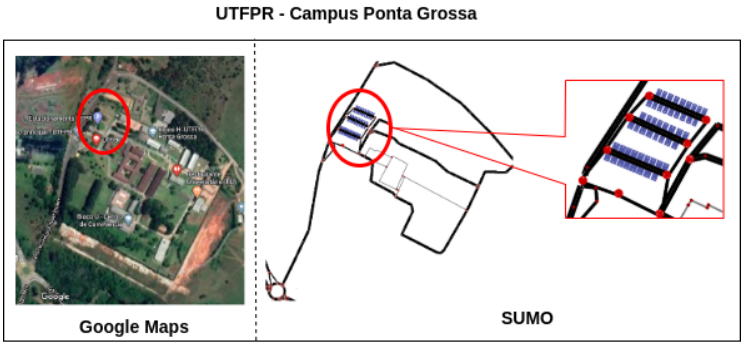

Figura 2. UTFPR (Ponta Grossa) - Mapa e Representação no SUMO

3.1. Camada dos Agentes

Ao todo são três grupos de agentes que compõem o SMA: *builder, pspace, driver*. Na Figura 3 são ilustrados os agentes, crenças e interações (mensagens).

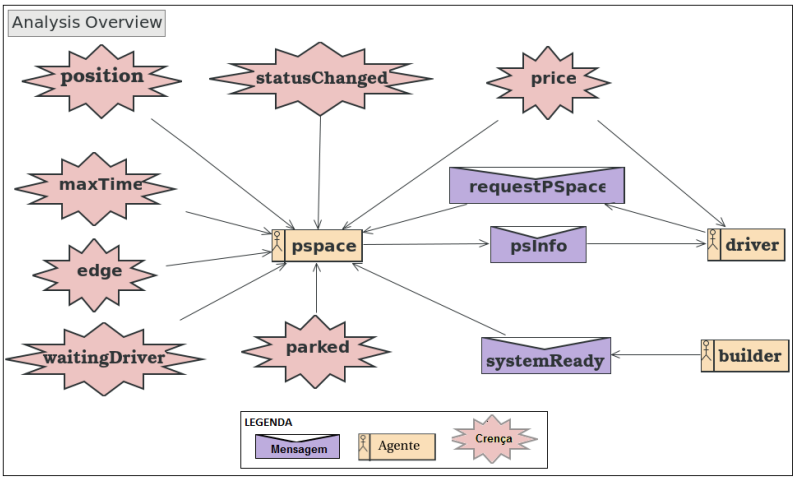

Figura 3. Diagrama Prometheus - Agentes: crenças e mensagens

O grupo de **agentes** possui os seguintes membros:

- *builder*: responsável pela instanciação de todos os agentes *pspace*, *workspaces* e artefatos do Cartago;
- *pspace*: responsável por representar a vaga do estacionamento, tanto de forma física, como virtual (no SUMO, conforme Fig. 2).
- *driver*: representante do motorista que deseja uma vaga de estacionamento.

O agente *pspace* possui as seguintes crenças:

- *price*: preço da vaga por hora utilizada;
- *statusChanged*: valor que corresponde a variação do sensor da vaga (em uso ou livre);
- *position*: posição da via da rede no SUMO (0 - ida / 1 - volta);
- *maxTime*: tempo máximo de permanência que um motorista pode utilizar a vaga. Caso esse tempo seja extrapolado, haverá alterações no valor de preço;
- *edge*: valor que identifica a aresta (rua) da rede do SUMO em que a vaga está posicionada. Por padrão o SUMO fornece os valores pré-estabelecidos como (edgez_ID), sendo esses editáveis;

- *waitingDriver*: valor que informa se a vaga está no status *aguardando motorista*. crença é utilizada no caso de alguma vaga ser ocupada sem ser previamente alocada a um determinado motorista. Assim, caso isso ocorra, a vaga foi ocupada de maneira indevida por algum objeto ou motorista não desejado;
- *parked*: valor que indica a presença de um motorista estacionado na vaga que foi alocada ao motorista.

Entre o grupo de agentes as seguintes mensagens podem ser trocadas:

- *requestPSpace*: mensagem enviada pelo agente *driver* ao agente *pspace* informando a requisição da vaga por um preço pré-determinado pela crença *price*.
- *systemReady*: mensagem enviada ao instanciar os agentes *pspace*, *workspaces* e artefatos do CArtAgO. Neste processo o agente *builder* envia uma mensagem aos agentes *pspace* informando que o sistema está em funcionamento. Assim, os agentes *pspace* estão abertos às requisições dos agentes *drivers*.
- *psInfo*: Quando o agente *pspace* aceita uma requisição pela vaga, esse agente deve informar ao agente respectivo *driver* as informações a respeito da vaga alocada.

Observa-se que está fora do escopo deste artigo detalhar o processo de requisição e negociação de uma vaga no estacionamento. Mas, existem trabalhos relacionados que apresentam soluções de SMA específicos para a negociação de vagas em estacionamentos inteligentes, como: [Castro et al. 2017], [Ducheiko et al. 2018] e [Alves et al. 2019].

3.2. Camada dos Artefatos - Conexão com SUMO

Os artefatos descritos nesse cenário compreendem a interligação do SMA com o SUMO, ilustrado na Fig. 4.

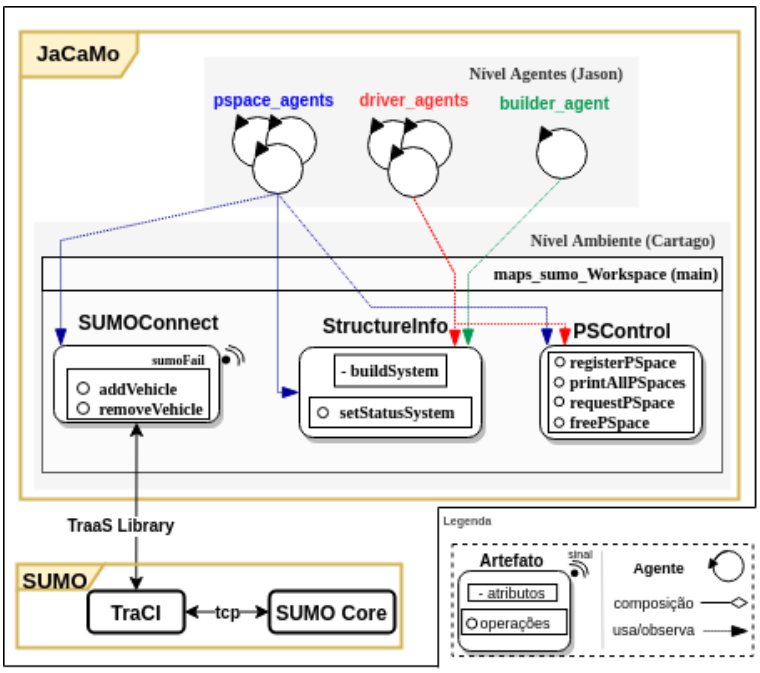

Figura 4. Interação entre JaCaMo e SUMO

Os seguintes artefatos são utilizados no sistema:

- ***SUMOConnect:*** Artefato utilizado para a conexão com o SUMO. A conexão com o SUMO é realizada por meio da biblioteca TraaS[2] [Krumnow 2013]. Há duas operações que os agentes *pspace* podem realizar no artefato *SUMOConnect*.
  - `addVehicle`: método que adiciona veículos na rede do SUMO. Os agentes *driver* do SMA em JaCaMo são considerados como veículos no SUMO, ao passo que os agentes *pspace* são os recursos *parkingArea* do SUMO. Listing 1 apresenta a função `addVehicle`. Os parâmetros da função são o nome do agente (id do veículo do SUMO), o id da estrada do SUMO (*edge*) e a posição da estrada (0 - ida, 1 - volta). A função inicialmente (linha 7) adiciona o veículo na rede do SUMO. Nessa linha ainda tem-se o `routeToParking`, que é previamente definido como uma sequência de estradas usadas pelo veículo para encontrar a vaga destinada a ele. Por fim, a linha 8 define qual vaga o veículo irá ocupar, sendo esta previamente negociada na camada dos agentes.

Listing 1. Função: `addVehicle` - CArtAgO

```
1 public class A_SUMOConnect extends Artifact {
2 SumoTraciConnection conn;
3 @OPERATION
4 public void addVehicle(String agentName, String edge, double position) {
5         conn.do_job_set(Vehicle.add(agentName, "DEFAULT_VEHTYPE", routeToParking,
          ↪  lane, position, speed, Byte.valueOf("0")));
6                 conn.do_job_set(Vehicle.setParkingAreaStop(agentName, parkingAreaName,
                  ↪  duration, timeOut, flag);}
```

  - `removeVehicle`: O Código Listing 2 apresenta a função de remoção do veículo da rede no SUMO. Após um agente *driver* finalizar a utilização de uma *pspace* é solicitado que o veículo correspondente no SUMO deixe a *parkingArea* (Linha 4) e inicie a rota para a saída da rede (routeToLeave) (Linha 5).

Listing 2. Função: `removeVehicle` - CArtAgO

```
1 @OPERATION
2 public void removeVehicle(String agentName) {
3                 conn.do_job_set(Vehicle.resume(agentName));
4                 conn.do_job_set(Vehicle.setRoute(agentName, routeToLeave));}
```

  Por fim, no *SUMOConnect* há a utilização do sinal `sumoFail` em caso de falha de conexão com o TraCI e problemas com a rede do SUMO.
- ***StructureInfo e PSControl:*** O artefato *PSControl* é utilizado para registro, alocação, liberação e controle das *pspaces*. Por sua vez, o artefato *StructureInfo* é utilizado pelo agente *builder* para informar aos demais agentes o status do SMA.

4. Integração: Agentes, Artefatos e o Meio Físico

Esta seção apresenta o JaCaMo como um sistema embarcado, o que proporciona a integração de um SMA no Meio Físico. Logo são apresentadas duas formas de conexão:

[2]Biblioteca baseada em Python que faz a comunicação com o TraCI do SUMO (*Socket* de conexão).

i. integração de um SMA a um meio físico utilizando diretamente a camada de agentes; e **ii.** integração do meio físico com a camada de ambiente (artefatos) de um SMA. Em ambas as abordagens, uma interface serial será utilizada como ferramenta de conexão com o meio físico. Esta interface é o Javino [Lazarin and Pantoja 2015], um protocolo de comunicação implementado em duas bibliotecas, sendo uma voltada para o *hardware* e a outra para o *software*. A principal característica desta interface serial é a confiabilidade, pois possui um processo de verificação da integridade da mensagem para que não ocorra perda ou qualquer tipo de alteração entre a mensagem enviada e a recebida.

4.1. ARGO: Integração de Agente Jason e Hardware

A integração de Agentes Jason com *hardware* permite desenvolver SMA em ambientes físicos reais. A noção de autonomia, proatividade e colaboração por parte dos agentes de um SMA justificam a necessidade de integrar este sistema a um ambiente real a fim de tratar as imprevisibilidade deste meio físico.

O ARGO [Pantoja et al. 2016] é uma arquitetura customizada de agentes para o *framework* Jason que acrescenta aos agentes a capacidade de controlar microcontroladores, por exemplo, Arduino. Os agentes com essa arquitetura são chamados de agentes ARGO e permitem desenvolver um SMA em um meio físico por meio de protótipos e qualquer dispositivos que utilizam microcontroladores. Os agentes ARGO possuem esta capacidade de controlar microcontroladores, pois na arquitetura deste agente foi incluída a interface de comunicação serial Javino.

Sendo assim, por meio da interface serial Javino, um agente ARGO pode enviar e receber informações de microcontroladores. As informações recebidas são tratadas como percepções que o agente ARGO recebe do ambiente via sensores do *hardware* que o agente está controlando e, portanto, são automaticamente adicionas como crenças na base de crenças deste agente. Já as informações enviadas para o microcontrolador são tratadas como ações que devem ser realizadas no meio físico via atuadores do *hardware*.

Para utilizar essas novas funcionalidades dos agentes ARGO, foram desenvolvidas quatro novas ações internas (*i.e.*, ações ou comportamentos pré-programados inerente ao agente [Bordini et al. 2007]) exclusivas para fazer o controle de microcontroladores, sendo elas:

- ***.port("Porta serial")***: Define qual porta serial o agente irá controlar e, consequentemente, qual microcontrolador o agente irá controlar. Esta ação interna possui um argumento que representa qual a porta serial o agente irá controlar.
- ***.perceive(open/block)***: Define se o agente irá perceber ou não o ambiente. Esta ação interna possui um argumento e tem somente duas opções de valores possíveis: *open*, que é utilizado para fazer o agente abrir as percepções e receber informações do *hardware*; e *block*, usada para fechar as percepções do agente e não mais receber informações.
- ***.limit(Tempo em milissegundos)***: Define um intervalo de tempo para o agente perceber o ambiente. Esta ação interna possui um argumento que representa o tempo em milissegundos que o agente irá alternar a percepção de aberta para fechada.
- ***.act("Ação")***: Define uma ação que o microcontrolador deve realizar. Esta ação interna possui um argumento que representa a ação que deverá ser enviada para o microcontrolador executar.

Além dos agentes ARGO, que permitem desenvolver SMA em ambientes físicos reais, existem também os Artefatos Físicos que permitem integrar artefatos convencionais do *framework* CArTAgO com dispositivos físicos e seus microcontroladores.

4.2. Artefatos Físicos: Integração CArtAgO e Hardware

A arquitetura proposta para o JaCaMo [Boissier et al. 2013] apresenta a diferença entre dois tipos de ambientes: o ambiente interno, aonde estão localizados os artefatos de um SMA, e o ambiente externo, que podem ser oferecidos por cenários simulados ou reais. Ao integrar a camada de ambiente do SMA com o meio físico, é possível desenvolver Sistemas Cyber-Físicos (SCF) usando artefatos.

O uso de artefatos para produzir SCF permite conectar o SMA e o meio físico. Portanto, os agentes do SMA não precisam incluir as responsabilidades de integração em seu ciclo de raciocínio. Consequentemente, esta abordagem é necessária para permitir que um agente trabalhe em um meio físico sem necessariamente ocupar todo o seu ciclo de raciocínio com os dados fornecidos por este meio. A vantagem da abordagem proposta é permitir que os agentes possam continuar interagindo com a camada de ambiente localizada dentro do SMA sem que eles precisem tomar conhecimento de que os artefatos estão ligados ao meio físico. O artefato responsável por esta integração com o meio físico é chamado Artefato Físico.

Um Artefato Físico é uma extensão de artefatos convencionais desenvolvida dentro do *framework* CArtAgO e portanto, também possui (1) um conjunto de operações que podem ser realizadas pelos agentes, (2) instruções que descrevem como estes artefatos devem ter suas funcionalidades acessadas, (3) propósito de existência, e (4) as estruturas internas que definem as implementações de suas funcionalidades [Ricci et al. 2006]. Com o CArtAgO, um artefato pode ser programado em linguagem Java, onde podem ser implementados métodos chamados de Operações, que determinam o comportamento do artefato, e também podem ser implementadas instâncias conhecidos como Propriedades Observáveis, que permitem ao artefato notificar o agente de algum evento do ambiente. Um artefato convencional do CArtAgO é considerado como um Artefato Físico quando ele troca mensagens com os dispositivos do meio físico, controlando os atuadores e monitorando os sensores deste dispositivo. Assim, um Artefato Físico pode fornecer ao agente os dados do meio físico através das Propriedades Observáveis, e os agentes podem realizar comandos que interfiram neste meio através das Operações.

A ponte de conexão entre a camada de ambiente e o meio físico é criada utilizando o Javino entre o Artefato Físico e o microcontrolador do dispositivo contido neste meio. Sendo assim, esta interface serial é incorporada como uma ferramenta de suporte aos Artefatos Físicos, possibilitando a comunicação artefato/dispositivo. Com isso, para implementar Artefatos Físicos em um projeto específico, três configurações devem ser feitas por três métodos abstratos:

- ***String definePort()***: retorna o valor da porta de comunicação a ser utilizada para trocar dados com o microcontrolador;
- ***int defineAttemptsAfterFailure()***: retorna o número de vezes que o artefato tentará enviar uma mesma mensagem quando algum erro tiver ocorrido durante a troca de mensagens;

- ***int defineWaitTimeout()***: retorna o tempo de espera em milissegundos entre dois pedidos feitos para o dispositivo;

Quando é necessário implementar as Operações e as Propriedades Observáveis dos Artefatos Físicos, dois métodos podem ser utilizados para realizar a comunicação com o dispositivo físico:

- ***String read()***: retorna uma mensagem enviada pelo microcontrolador do dispositivo, normalmente apresentando as medidas coletadas pelos seus recursos.
- ***void send(String message)***: envia a mensagem passada por parâmetro ao dispositivo, normalmente apresentando um comando para operar os recursos deste dispositivo.

A abordagem de Artefatos Físicos permite que a conexão entre o SMA e o meio físico não ultrapasse o Nível de Ambiente proposto, e com isso, o acoplamento entre as tecnologias é diminuído.

5. Integração: Agentes e Aplicações IoT

Uma vez que SMAs, dispositivos, plataforma e ferramentas IoT possuem diferentes escopos e implementações, é necessário então a utilização de diferentes tecnologias que realizem a interface de comunicação entre eles.

A integração desenvolvida nesse trabalho utiliza diferentes módulos e conexões para essa interface de comunicação seja realizada. A Figura 5 apresenta a integração, onde os módulos, bem como suas conexões e interações sequenciais são descritas abaixo:

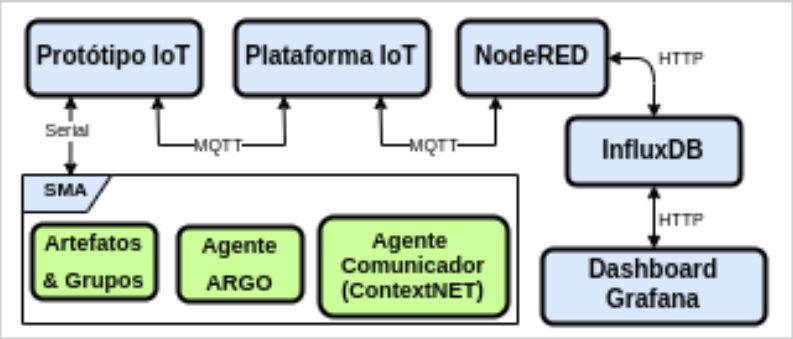

Figura 5. Integração: Protótipo, SMA, plataforma e ferramentas IoT

- **Protótipo IoT:** O protótipo é composto por três itens: NodeMCU 1.0 (ESP 8266), Arduino UNO e um Raspberry Pi 3. Inicialmente o Arduino realiza a leitura e envia os dados dos sensores (ultrasônico de distância) via serial para o NodeMCU. Assim, o NodeMCU gera um pacote JSON e envia via MQTT para a Plataforma IoT. O Raspberry Pi 3 é utilizado na hospedagem dos módulos do NodeRED, InfluxDB e Grafana;
- **SMA:** O SMA é composto por artefatos (desenvolvidos no Cartago), grupos sociais (providos pelo Moise) e agentes, sendo eles:
  - Agente ARGO: Estabelece a conexão com o protótipo IoT (Arduino UNO) para a percepção do ambiente físico por meio dos sensores ultrasônicos de distância. Ao obter as percepções, o agente ARGO notifica o protótipo IoT (NodeMCU) e o agente Comunicador caso ocorram mudanças no ambiente (alteração significativa nos valores dos sensores).

  - Agente Comunicador (ContextNET): Agente responsável pela comunicação com outros SMAs via rede ContextNET. Com a utilização desse agente, o SMA é capaz de se comunicar com outros SMAs com a finalidade de uma múltipla cooperação entre os sistemas. Por exemplo, uma rede de estacionamentos inteligentes disponibilizados em vários locais de uma mesma cidade. Assim, um único estacionamento poderá prover vagas de n-estacionamentos.
- **Plataforma IoT:** A plataforma IoT visa a integração entre diferentes dispositivos IoT por meio de rotas MQTT (publish/subscribe). Ao receber as mensagens via MQTT do protótipo, a plataforma redireciona as mensagens via MQTT para o NodeRED. A plataforma utilizada é a fornecida pela KonkerLabs, contudo há outras soluções fornecidas pela Eclipse, Google, entre outros;
- **NodeRED:** Utilizado para a decomposição e extração das mensagens JSON provindas da plataforma IoT e armazenagem no InfluxDB via HTTP. A decomposição e extração é utilizada para a obtenção do valor dos sensores do protótipo;
- **InfluxDB:** Banco de dados de séries de tempo *(time series databases - TSDB)*. A utilização desse tipo de banco de dados é devido a sua característica de ter como chave primária um *timestamp*. Assim, a armazenagem dos valores dos sensores é efetuada de acordo com o tempo de medição;
- ***Dashboard* Grafana:** O *dashboard* do Grafana provê diferentes componentes para a visualização de dados em tempo real. O dashboard acessa os dados dos sensores via HTTP no banco de dados do InfluxDB.

6. Conclusão

Este trabalho apresenta uma abordagem para integração de um SMA desenvolvido em JaCaMo com ferramentas de Simulação Urbana, Sistemas Embarcados e a camada IoT. O objetivo principal não é apenas descrever tal integração, mas também não estabelecer um forte acoplamento entre os níveis de ambiente, agentes, IoT e simulação.

A vantagem desta abordagem está na sua generalização. Ou seja, os níveis propostos são heterogêneos, e portanto permitem que diferentes aplicações utilizem esta abordagem como suporte para integração de sistemas. Além disso, outra vantagem está no baixo acoplamento entre as camadas, o que dá flexibilidade para utilizar tanto a arquitetura completamente integrada, como apenas partes dela. Através deste trabalho problemas que demandem escalabilidade e um sistema inteligente podem ser resolvidos com auxílio da IoT integrada a um SMA. Ainda é possível definir se o tipo de ambiente a ser utilizado será simulado ou físico.

Como trabalhos futuros, busca-se desenvolver uma metodologia de integração como um *framework* para soluções de mobilidade urbana integrado SMA, IoT, Simulação e Sistemas Embarcados. Além disso, é necessário implantar uma aplicação (mais robusta e complexa) no domínio de Cidades Inteligentes, a qual utilize todos os níveis propostos. Com tal aplicação igualmente será possível executar experimentos para avaliar características da metodologia de integração, como: escalabilidade, baixo acoplamento, confiabilidade na troca de mensagens e informações, entre outras. Por fim, pretende-se contemplar a integração da camada de organização social com o Moise (JaCaMo). Assim, será possível estabelecer regras e comportamentos em sociedades de agentes, promovendo a organização social e a criação de grupos sociais de agentes, esses elementos podem representar de forma adequada características da mobilidade urbana.

Referências

Albino, V., Berardi, U., and Dangelico, R. M. (2015). Smart cities: Definitions, dimensions, performance, and initiatives. *Journal of urban technology*, 22(1):3–21.

Alves, B. R., Alves, G. V., Borges, A. P., and Leitão, P. (2019). Experimentation of Negotiation Protocols for Consensus Problems in Smart Parking Systems. In Marik, V., Kadera, P., Rzevski, G., Zoitl, A., Anderst-Kotsis, G., Tjoa, A. M., and Khalil, I., editors, *Industrial Applications of Holonic and Multi-Agent Systems*, Lecture Notes in Computer Science, pages 189–202, Cham. Springer International Publishing.

Boissier, O., Bordini, R. H., Hübner, J. F., Ricci, A., and Santi, A. (2013). Multi-agent oriented programming with jacamo. *Science of Computer Programming*, 78(6):747–761.

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

Castro, L. F. S. D., Alves, G. V., and Borges, A. P. (2017). Using trust degree for agents in order to assign spots in a Smart Parking. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, 6(2):45–55.

Ducheiko, F. F., André, P. B., and Gleifer, V. A. (2018). Implementação de Modelo de Raciocínio e Protocolo de Negociação para um Estacionamento Inteligente com Mecanismo de Negociação Descentralizado. *Revista Junior de Iniciação Científica em Ciências Exatas e Engenharia*, 1(19):25–32.

Evans, D. (2011). How the Next Evolution of the Internet of Things Is Changing Everything. *Cisco Internet Business Solutions Group*, page 11.

Krajzewicz, D., Hertkorn, G., Rössel, C., and Wagner, P. (2002). Sumo (simulation of urban mobility)-an open-source traffic simulation. In *Proceedings of the 4th middle East Symposium on Simulation and Modelling (MESM20002)*, pages 183–187.

Krumnow, M. (2013). Sumo as a service–building up a web service to interact with sumo. In *Simulation of Urban MObility User Conference*, pages 62–70. Springer.

Lazarin, N. M. and Pantoja, C. E. (2015). A Robotic-Agent Platform for Embedding Software Agents using Raspberry Pi and Arduino Boards. In 9^{th} *Software Agents, Environments and Applications School*.

Neirotti, P., Marco, A. D., Cagliano, A. C., Mangano, G., and Scorrano, F. (2014). Current trends in smart city initiatives: Some stylised facts. *Cities*, 38:25 – 36.

Pantoja, C. E., Stabile Jr, M. F., Lazarin, N. M., and Sichman, J. S. (2016). Argo: A customized jason architecture for programming embedded robotic agents. *Fourth International Workshop on Engineering Multi-Agent Systems (EMAS 2016)*.

Ricci, A., Viroli, M., and Omicini, A. (2006). Programming MAS with artifacts. 3862 LNAI:206–221.

Wooldridge, M. J. (2000). *Reasoning about rational agents*. MIT press.