Uma abordagem multiagente para o controle de carros inteligentes autoadaptativos

Clebson Uchoa , Rendley rnou[2], Marcos de Oliveira[3]

Universidade Federal do Ceará - Campus Quixadá
v. José de Freitas Queiroz, 5003 - Cedro, Quixadá - CE, 63900-000

clebson@ lu.ufc.br, rendley rnou@ lu.ufc.br, m rcos.oliveir @ufc.br

bstract. *To build systems for autonomous control it is necessary to consider reactive situations, where the system reacts to changes in the environment and, especially, situations of change, where the system needs to adapt its reactions to new behaviors observed in the environment. This work aims to implement a smart car system using the multi-agent paradigm, where the car system is responsible for self-adapting, modifying its environment according to the preferences of the user who drives the vehicle.*

Resumo. *Para construção de sistemas para controle autônomo é preciso considerar situações reativas, onde o sistema reage a alterações no ambiente e, principalmente, situações de mudanças, onde o sistema precisa adaptar as suas reações a novos comportamentos observados no ambiente. Este trabalho tem como objetivo implementar um sistema de um carro inteligente utilizando o paradigma multiagente, onde o sistema do carro é responsável por se autoadaptar, modificando seu ambiente de acordo com as preferências do usuário que dirige o veículo.*

1. Introdução

No desenvolvimento de software utilizando o paradigma multiagente, um agente é um sistema informático situado em um ambiente que é capaz de realizar ações de forma autônoma para conseguir seus objetivos [Wooldridge 2009].

Este artigo aborda a implementação de um sistema de um carro inteligente utilizando o paradigma multiagente, onde o sistema do carro é responsável por se autoadaptar, modificando seu ambiente de acordo com o usuário que dirige o veículo.

pós a descrição do problema a ser abordado é criada uma modelagem de sistema, que é utilizado para apresentar uma perspectiva geral do sistema, descrevendo as responsabilidades e fluxo de execução dos módulos do projeto. Em seguida o sistema é desenvolvido a partir da modelagem.

O artigo é dividido nas seguintes seções: Fundamentação teórica, onde são definidos os seguintes conceitos: sistemas multiagente, computação ubíqua, sistemas autoadaptativos, e a linguagem de programação S RL. Especificação do ambiente e contexto, arquitetura e construção do sistema, funcionamento do sistema multiagente e conclusão.

2. Fundamentação teória

Esta seção apresentará os principais conceitos relacionados a este trabalho. Na seção 2.1 é apresentado o conceito de Sistemas multiagente, na seção 2.2 o conceito de Computação

ubíqua, na seção 2.3 o conceito de Sistemas autoadaptativos e por fim na seção 2.4 é demonstrado o conceito da linguagem de programação S RL.

2.1. Sistemas multiagente

Sistemas multiagentes são sistemas compostos por múltiplos agentes, que exibem um comportamento autônomo, mas ao mesmo tempo interagem entre si no âmbito do sistema [Reis 2019]. Segundo [Zambonelli et al. 2000], os sistemas multiagentes são divididos em duas classes:

Sistemas de Resolução Distribuída de Problemas: Os agentes cooperam entre si para atingir um objetivo comum, todos os agentes se conhecem e confiam uns nos outros.

Sistemas bertos: Não existe um objetivo comum para os agentes, e sim para o sistema, podendo assim qualquer agente entrar e sair do sistema. Essa liberdade dos agentes pode acarretar em prejuízos ao sistema, com a aquisição de agentes com objetivos diferentes aos do sistema.

Os agentes de um sistema podem ser coordenados para atingir um objetivo comum a todos os integrantes, dividindo assim o problema em problemas menores, para que cada agente possa trabalhar com objetivos específicos e com menos complexidade.

2.2. Computação ubíqua

O conceito mais abrangente de Computação ubíqua ou pervasiva é o que a define como invisível a olho nu, mas sabe-se que ela está presente no espaço [Wang 2011]. Como exemplo de Computação Pervasiva pode-se citar uma casa inteligente, onde vários sensores atuam integrados para que se tenha as condições desejadas no ambiente. Computação Móvel pode ser compreendida como o acesso a informação em qualquer lugar e momento. Para que isso aconteça há uma diversidade de equipamentos como os celulares, tablets e navegadores. [Ladd et al. 2010].

fusão dessas tecnologias resulta na Computação Ubíqua que possui como suas principais características a descentralização, diversidade e a conectividade.

2.3. Sistemas autoadaptativos

Segundo [Oreizy et al. 1999], sistemas autoadaptativos modificam o próprio comportamento em resposta a mudanças no seu ambiente operacional. Pelo ambiente operacional entende-se qualquer coisa observável pelo sistema de software, tais como entradas de usuário, dispositivos externos de hardware e sensores. O ponto chave dos softwares autoadaptativos é que o ciclo de vida do software nunca termina, pois o software precisa estar constantemente se avaliando e alterando.

Esses sistemas podem ser definidos como sistemas que correspondem as mudanças de ambiente podendo alterar o seu estado ou comportamento de acordo com elas [Sawyer et al. 2010]. Vale salientar que um sistema autoadaptativo é ciente de contexto, ou seja, utiliza seus próprios recursos (hardware ou software) para o conhecimento da intenção do usuário.

2.4. S RL

S RL é uma linguagem de programação para o desenvolvimento de sistemas multiagentes, possui diversas características que facilitam o desenvolvimento e o entendimento

de um sistema desse paradigma. É independente de arquitetura, projetada para facilitar a aplicação de conceitos fornecidos por outros modelos, possui um conjunto reduzido de conceitos-chave que facilitam o entendimento, concentrando-se exclusivamente nos princípios essenciais para a implementação de um sistema multiagente. Um sistema S RL é programado como um conjunto de agentes que interagem em um conjunto de espaços. linguagem possui interoperabilidade com java.

S RL é baseada em quatro conceitos principais: gente, Capacidade, Espaço e habilidade:

gente: Possui um conjunto de capacidades que descrevem o que ele é capaz de realizar.
Capacidade: É a especificação de um conjunto de ações, que não faz suposições de implementação. Dentre as várias capacidades de um agente, a capacidade do "comportamento" determina a conduta do agente.
Espaço: Os estados mentais de um agente são definidos como atributos dentro do agente criado.
Habilidade: É a possível implementação de uma capacidade.

No contexto do carro inteligente, cada agente posui seu estado mental (crenças), e é responsável por realizar uma tarefa implementatada como uma habilidade com base nas informações que possui.

3. Especificação do ambiente e contexto

Para a construção de sistemas inteligentes é preciso considerar situações onde o sistema possa interagir com o ambiente, percebendo qual o estado ele se encontra e reagindo a alterações no ambiente, onde o sistema precisa se adaptar a novos comportamentos observados.

solução apresentada neste trabalho tem como objetivo fazer com que o sistema permita que os dados de preferência de um usuario sejam utilizados pelo carro atual e por outros carros, permitindo que o usuário tenha suas necessidades atendidas a partir da adaptação com base nas suas preferências. Deste modo o sistema é responsável por se autoadaptar, modificando seu ambiente de acordo com o usuário que dirige o veículo.

Foram utilizadas ferramentas para o desenvolvimento de sistemas multiagentes, sendo uma delas a linguagem de programação S RL, que visa fornecer as abstrações fundamentais para lidar com concorrência, distribuição, interação, descentralização, reatividade, autonomia e reconfiguração dinâmica.

4. rquitetura e construção do sistema

O sistema desenvolvido possui módulos distribuídos e responsabilidades bem definidas, onde as funcionalidades são realizadas de maneira independente por cada módulo. O que permite melhor entendimento do fluxo de execução do sistema, facilidade na manutenção dos módulos e implementações de futuras funcionalidades.

4.1. plicativo móvel

O aplicativo móvel desenvolvido com o framework Flutter [1]permite ao usuário registrar-se no sistema, onde é necessário informar o nome de usuário e as preferência iniciais do

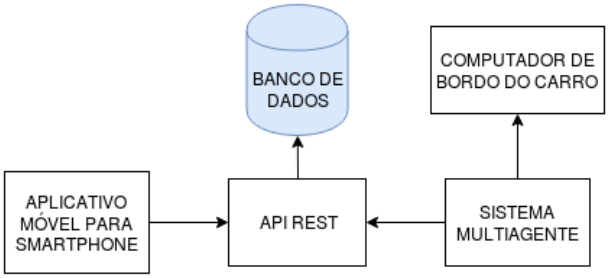

Figura 1. rquitetura dos módulos do sistema para controle do carro inteligente.

carro, como: inclinação do banco e altura do volante. Tais informações são codificadas em um arquivo json que é enviado para a PI REST por meio do método POST do protocolo HTTP.

4.2. PI REST

PI REST desenvolvida com o framework Spring Boot[2]possui *endpoints* para permitir a persistência e consulta de dados de preferências de usuários, a PI persiste os dados em um banco de dados PostgreSQL. O sistema multiagente pode editar as informações de preferência de um determinado usuário e consultá-las a partir do nome de usuário para utilizá-las na adaptação do carro.

4.3. Sistema multiagente

O sistema multiagente desenvolvido com a linguagem S RL [Rodriguez et al. 2014] é responsável por identificar qual usuário está no interior do carro, busar as preferências referentes a esse usuário na PI e enviá-las para o computador de bordo do carro (implementado apenas com simulações), para que sejam aplicadas ao ambiente do automóvel. lém disso, é responsável por verificar eventuais mudanças feitas no interior do veículo pelo usuário (mudanças na inclinação no banco ou altura do volate por exemplo) e enviar tais mudanças para a PI, para que sejam persistidas e utilizadas futuramente quando o usuário entrar novamente no carro.

5. Funcionamento do sistema multiagente

O sistema multiagente desenvolvido possui três agentes principais: o agente adaptador de carro, o agente verificador de mudanças e agente *boot*.

Na linguagem S RL, a comunicação entre os agentes ocorre por meio da emissão e percepção de eventos. O agente *boot* possui a responsabilidade de inicializar todos os agentes que compõem o sistema multiagente e de realizar a simulações de situações reais que ocorrem dentro do carro, como a presença de um novo usuário e modificações no ambiente do carro, tais simulações são implementadas como eventos que são definidos como mensagens enviadas que podem ou não conter atributos [Rodriguez et al. 2014].

Para simular a presença de um novo usuário no carro, o agente *boot* emite o evento ”UserDetected”que recebe como parâmetro o nome do usuário, o evento ”UserDeteced”é

[1]Site do Flutter: https://flutter.dev/
[2]Site do Spring Boot: https://spring.io/projects/spring-boot

detectado pelo agente ”Car dapter gent”, que utiliza a informação do nome do usuário levado pelo evento para buscar as preferências desse usuário na PI e enviá-las para o computador de bordo para serem aplicadas ao carro.

De forma semelhante, o agente *boot* emite um evento com um atributo informando o estado atual do carro para simular a mudança que ocorreu em alguma característica interna do carro, tal evento é detectado pelo agente ”ChangeCheker gent”, que envia as novas informações de preferência oriundas do evento para a PI REST, responsável por persistir a atualização das informações de preferência do usuário, para que os novos dados possam ser utilizadas futuramente.

6. Conclusões

O sistema comportou-se como o esperado, reagindo aos eventos simulados e realizando as funções e fluxos de ações necessárias para que a ideia proposta do funcionamento do carro inteligente autoadaptativo pudesse acontecer. ideia de um modelo distribuido com diferentes módulos se mostrou a mais adequada em um mundo onde cada vez mais os dispositivos estão conectados e compartilhando informações.

Este artigo reforça ainda mais o reconhecimento, a importância, e a relação entre as diferentes tecnologias para apoiar o desenvolvimento de sistemas multiagente que trazem benefícios à sociedade. uxiliando a identificar as atitudes a serem tomadas para modelar sistemas multiagentes desse tipo, e consequentemente facilitando seu desenvolvimento.

Referências

Ladd, D., Datta, ., Sarker, S., and Yu, Y. (2010). Trends in mobile computing within the is discipline: ten-year retrospective. *Communications of the ssociation for Information Systems*, 27.

Oreizy, P., Gorlick, M., Taylor, R., Heimhigner, D., Johnson, G., Medvidovic, N., Quilici, ., Rosenblum, D., and Wolf, . (1999). n architecture-based approach to self-adaptive software. *Intelligent Systems and their pplications, IEEE*, 14:54 – 62.

Reis, L. (2019). Coordenação em sistemas multi-agente : plicações na gestão universitária e futebol robótico.

Rodriguez, S., Gaud, N., and Galland, S. (2014). Sarl: general-purpose agent-oriented programming language.

Sawyer, P., Bencomo, N., Whittle, J., Letier, E., and Finkelstein, . (2010). Requirements-aware systems: research agenda for re for self-adaptive systems. In *2010 18th IEEE International Requirements Engineering Conference*, pages 95–103.

Wang, J.-Y. (2011). cost-effective rfid encoding method for inventory identification. *frican Journal of Business Management*, 5.

Wooldridge, N. M. (2009). *n Introduction to Multiagent Systems*, volume 2. Wiley.

Zambonelli, F., Jennings, N., and Wooldridge, M. (2000). Organisational abstractions for the analysis and design of multi-agent systems.