Protótipo de interface Web para o framework MASPY

Leonardo B. Lopes[1], Rafael C. Cardoso[2], André P. Borges[1], Gleifer V. Alves[1]

[1]Universidade Tecnológica Federal do Paraná (UTFPR)
Ponta Grossa, PR, Brasil

[2]University of Aberdeen
Aberdeen, United Kingdom

leonardoborgeslopes@alunos.utfpr.edu.br, rafael.cardoso@abdn.ac.uk

{apborges,gleifer}@utfpr.edu.br

Abstract. *MASPY is a framework used for the development of multiagent systems using the BDI architecture. MASPY is based on a distributed system where agents exchange several messages. So, we realise the need for a web interface to visualize the execution of the system. With this, our main goal is to present an interface prototype for MASPY.*

Resumo. *MASPY é um framework em desenvolvimento para sistemas multiagentes utilizando a arquitetura BDI. MASPY é baseado em um sistema distribuído onde os agentes trocam inúmeras mensagens. Logo, durante seu desenvolvimento surge a necessidade de uma interface web para auxiliar na visualização dos seus componentes para que seja possível acompanhar a execução do sistema. Com isso, objetivo principal deste trabalho é apresentar um protótipo de interface para o MASPY.*

1. Introdução

Sistemas Multiagentes (do inglês *MultiAgent System* - MAS) é uma área de estudo onde há vários agentes autônomos que podem estar atuando de forma cooperativa ou independente [Wooldridge 2009]. É possível ter agentes com diferentes objetivos e planos atuando em diversos contextos, porém todos em busca de resolver um ou mais problemas em questão. Esta abordagem facilita a resolução de problemas complexos, permitindo delegar funções entre os agentes por meio da divisão de tarefas [Wooldridge 2009].

O paradigma *Belief, Desire and Intention* (BDI) é usado para representar o conhecimento no desenvolvimento de agentes e sistemas autônomos, modelando a tomada de decisão e os comportamentos dos agentes por meio de Crenças, Desejos e Intenções, como foi aprofundado em [Bratman 1987].

O framework MASPY surgiu como uma alternativa para facilitar o desenvolvimento de MAS utilizando o paradigma BDI, com o seu diferencial sendo a implementação em Python, diferente da maioria dos outros frameworks [Mellado et al. 2023]. Nesse contexto, observou-se a necessidade de uma interface gráfica para facilitar a visualização do sistema e a interação com os usuários.

Durante a investigação, foram analisados outros frameworks semelhantes, como JaCaMo [Boissier et al. 2020], JADE [Bellifemine et al. 1999] e Spade-BDI [Palanca et al. 2020]. Neles, foram estudados os recursos de suas interfaces gráficas

e quais características seriam úteis para o MASPY. Com os frameworks analisados, foram elaborados os primeiros protótipos de telas da interface, apresentados neste artigo.

2. Trabalhos Relacionados

Os frameworks analisados foram escolhidos devido às suas funcionalidades específicas e visando serem implementados no MASPY.

JaCaMo[1] é um framework voltado para o desenvolvimento de MAS. Na camada Moise, ele utiliza o conceito de organizações para coordenar agentes em torno de um objetivo; ainda opera com agentes BDI através do Jason e com o conceito de ambiente na CArtAgO. Nesse cenário, cada agente vai ter três especificações, uma estrutural que define o papel de cada agente, uma funcional que planeja como os objetivos serão atingidos, e uma normativa que define as obrigações de cada papel e missão [Boissier et al. 2020]. Conta ainda com uma funcionalidade interessante, o *Mind Inspector*, que permite visualizar em uma interface Web as características de cada organização e agente, com um diagrama do sistema e os desejos, crenças e intenções

JADE[2] também é voltado para facilitar o desenvolvimento de MAS, porém ele segue as diretrizes da *Foundation for Intelligent Physical Agents* (FIPA). Para alcançar isso, ele conta com várias ferramentas: uma plataforma para o gerenciamento dos agentes, agentes distribuídos, interface de programação, transporte de mensagens entre os agentes e interface gráfica para gerenciar diversos agentes e plataformas de agente do mesmo agente [Bellifemine et al. 1999]. Através da interface, é possível criar e visualizar agentes, além de enviar mensagens.

Spade-BDI[3] é voltado para o desenvolvimento de MAS utilizando a lógica BDI, nele cada agente conta com uma interface web onde é possível se conectar identificando seu servidor e senha. A partir disso, é carregado um dashboard contendo as informações dos comportamentos e os contatos (agentes). Cada comportamento, ao ser selecionado, exibe a caixa de mensagens utilizada entre os agentes e um diagrama de estados finitos. Além disso, os contatos também podem ser clicados para exibir algumas informações do agente, como as mensagens e o status [Palanca et al. 2020].

Baseado nesses trabalhos, o protótipo incluirá um dashboard inspirado no Spade-BDI, a apresentação de todos os agentes e sua modelagem como no JADE e no *Mind Inspector* do JaCaMo. Inspirando-se em soluções existentes buscamos atender as demandas do MASPY e superar os limites individuais dos outros trabalhos, tornando a visualização mais amigável, do que o JaCamo e JADE por exemplo, e atribuindo um foco no sistema como um todo e não em cada agente específico como observado no Spade-BDI.

3. Fundamentação Teórica

Os agentes BDI possuem características próprias: os *Beliefs* (Crenças) se referem ao conhecimento do agente, as crenças em que ele acredita; os *Desires* (Desejos) significam o que o agente pretende fazer, os seus objetivos; e as *Intentions* (Intenções) são os planos pertencentes ao agente para atingir determinado objetivo [Bratman 1987].

[1]https://jacamo-lang.github.io/ acessado em 17/06/2025

[2]https://jade.tilab.com/, acessado em 19/07/2025

[3]https://spade-mas.readthedocs.io/en/latest/, acessado em 14/07/2025

MASPY abstraiu os conceitos de BDI através do uso de *Beliefs* (Crenças), *Objectives* (Objetivos) e *Plans* (Planos). Crenças e Objetivos podem conter uma chave e um valor para armazenar qualquer forma de dado, já os Planos são construídos através de algo que irá causar a sua ativação, o contexto para ativá-lo e um corpo contendo as Crenças ou Objetivos que o agente precisa possuir para executar o plano [Mellado et al. 2023].

Além disso, utiliza-se do conceito de *Environment* (Ambiente) que pode ser interpretado como uma entidade onde os agentes podem estar situados, por exemplo: uma rua ou uma sala. Com esse conceito, os agentes conseguem perceber e agir de diferentes maneiras conforme o ambiente em que estão situados. Junto a isso, existe o conceito dos *Facts* (Fatos). Para o ambiente, Fatos são semelhantes ao que as Crenças são para os agentes, porém há uma diferença em quais agentes podem alterar esses Fatos.

MASPY conta também com a comunicação entre os agentes via mensagens transmitidas por um tópico. Os agentes só conseguem se comunicar com outros agentes inscritos no mesmo tópico, porém todos os agentes são conectados a um tópico sem nome conhecido por *default*. O protocolo da mensagem é dividido em cinco parâmetros: remetente, destino, o tipo (ou ação) da mensagem, conteúdo e o tópico a ser usado (parâmetro opcional). As diretivas definem qual será o tipo da mensagem e são usadas principalmente para a troca de Crenças ou Objetivos. Os agentes podem informar, requisitar ou pedir quais são os conteúdos dos outros agentes.

Com as características mencionadas, foi definido que a interface será implementada em Python, devido à proximidade com o MASPY, e para o desenvolvimento dos protótipos foi usada a ferramenta Figma[4].

4. Metodologia

No decorrer do uso do MASPY observou-se a necessidade de alguns recursos para a interface Web. A partir disso, foi feito um levantamento de requisitos que apontou a necessidade das seguintes funcionalidades:

- Dashboard contendo as informações gerais do sistema, como os números de agentes, planos, ambientes, agentes ativos, objetivos e histórico com o total de mensagens no decorrer do processamento.
- Interface para exibir os agentes e suas informações.
- Interface para exibir a troca de mensagens entre os agentes.

Na listagem dos agentes, notou-se a importância de um filtro que permita selecionar quais agentes serão mostrados. Dentro disso, cada agente deve exibir suas informações (*name, belief, goal, plans, topics e environments*), isso será feito por meio de um pop-up que será aberto ao clicar no agente. Como *plans, topics e environments* podem ter múltiplos itens, a exibição é feita por meio de um *dropdown*. Como é voltado para um sistema MAS, essa tela ainda incluirá uma opção de paginação na mesma seção que o filtro, nela podem ser selecionados quantos agentes serão exibidos na página e ainda trocar de página.

Já na listagem das mensagens, a exibição foi pensada no formato de um diagrama de mensagens que incrementa na parte inferior da tela. Essa tela conta com duas caixas identificadas como *sender e target*, uma flecha sai do *sender* e vai em direção ao *target*.

[4]https://www.figma.com/

Para revelar as informações da mensagem pode-se clicar em uma flecha onde será aberto um pop-up contendo: destinatário, remetente, tipo da mensagem, tópico e conteúdo.

5. Resultados

Nesta seção são apresentadas as telas iniciais desenvolvidas. Conforme mencionado anteriormente todas as telas apresentam uma seção de navegação na parte superior com os links para as telas *Home, Agents* e *Messages*. Na tela inicial foi implementado um dashboard contendo os números totais de agentes, crenças, planos, mensagens enviadas, objetivos e ambientes. Abaixo disso há duas seções, agentes e mensagens, que mostrarão informações diferentes ao serem clicadas.

A seção dos agentes, Figura 1, contém as informações gerais dos agentes como: agentes inscritos em qualquer tópico (desconsiderando o *default*), usados no processo, sem crenças, sem objetivos, sem planos, que não estão em um ambiente, número de fatos e de agentes com fatos.

A seção das mensagens, Figura 2, mostra o número de tópicos cadastrados no sistema, tópicos usados no processo e a quantidade de mensagens enviadas por cada tópico.

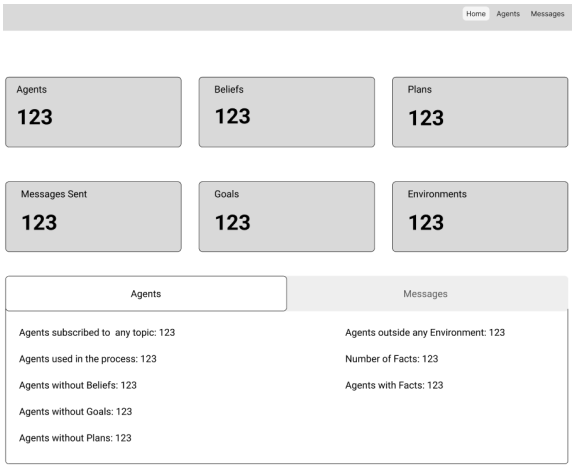

Figura 1. Tela Home (Agents)

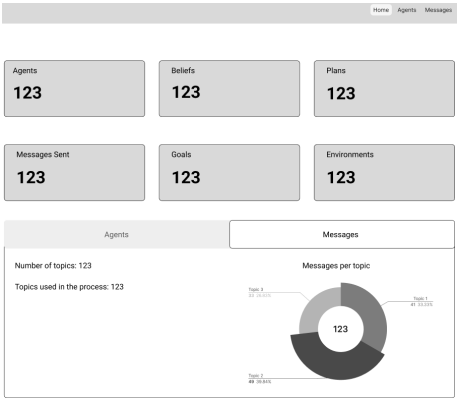

Figura 2. Tela Home (Messages)

Essa tela contém uma seção ao lado esquerdo que permite filtrar os agentes a serem mostrados e a principal característica que são os agentes do sistema sendo exibidos, como mostrado na Figura 3. Cada agente, ao ser clicado, abre um pop-up (Figura 4) contendo as informações: *name, belief, goal, plans, topics e environments*.

A tela de mensagens, evidenciada na Figura 5, possui uma seção à esquerda destinada ao filtro por agente, onde pode selecionar qual agente é o *sender* e/ou o *target*, e exibe duas caixas, uma intitulada como *Sender* e a outra *Target*. Abaixo delas, observa-se um gráfico com o histórico de mensagens enviadas através de uma seta. Cada seta contém o nome do agente que enviou a mensagem (*Sender*) e quem recebeu ela (*Target*), ao serem clicadas um *pop-up* (Figura 6) é exibido com as informações de cada mensagem: *sender, target, action, topic e content.*

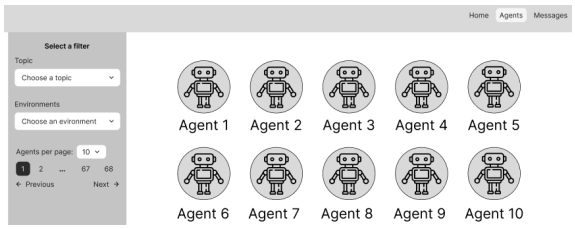

Figura 3. Tela Agents

Figura 4. Popup do agente

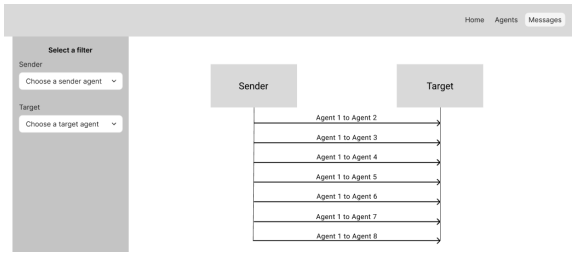

Figura 5. Tela Messages

Figura 6. Popup da mensagem

6. Conclusão

Com as informações demonstradas, fica clara a importância de uma interface Web para auxiliar no desenvolvimento de MAS utilizando o framework MASPY. O principal ponto da interface é a exibição das características de agentes e mensagens no sistema, sendo efetivas como meios de depuração do código. Porém, junto com essas finalidades vêm os desafios de implementação, sendo o principal deles como passar as informações do código para a interface; um exemplo para resolver isso está no Spade-BDI, que utiliza o protocolo XMPP [Palanca et al. 2020]. Essa parte do projeto ainda requer estudo e está nos planos futuros.

Outro desafio é a exibição de um conjunto enorme de informações como o de um MAS. Neste protótipo, uma estratégia adotada foram as paginações e os *dropdowns*, onde pode-se obter versatilidade quando as informações forem exibidas; entretanto, ainda há espaço para melhorias.

Dentro dos passos futuros também está presente o estudo do ChronIDE, um framework Web voltado para o desenvolvimento de sistemas embarcados de agentes [Souza de Jesus et al. 2023], a validação de usabilidade do protótipo, prevista para ser feita através de um questionário, em um grupo selecionado de indivíduos, abordando os principais pontos do projeto. Esse questionário ajudará a levantar novos requisitos e, assim, melhorar e implantar o desenvolvimento da interface.

Agradecimentos: Este trabalho tem financiamento do projeto: 444568/2024-7, CNPq/MCTI/FNDCT Nº 22/2024, *Programa Conhecimento Brasil – Apoio a Projetos em Rede com Pesquisadores Brasileiros no Exterior.*

Referências

Bellifemine, F., Poggi, A., and Rimassa, G. (1999). Jade - a fipa-compliant agent framework.

Boissier, O., Bordini, R. H., Hübner, J. F., and Ricci, A. (2020). *Multi-Agent Oriented Programming: Programming Multi-Agent Systems Using JaCaMo*. Intelligent Robotics and Autonomous Agents. MIT Press.

Bratman, M. (1987). *Intention, Plans, and Practical Reason*. Cambridge, MA: Harvard University Press, Cambridge.

Mellado, A., Fidler, I., Borges, A., and Alves, G. (2023). Maspy: Towards the creation of bdi multi-agent systems. In *Anais do XVII Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações*, pages 106–117, Porto Alegre, RS, Brasil. SBC.

Palanca, J., Terrasa, A., Julian, V., and Carrascosa, C. (2020). Spade 3: Supporting the new generation of multi-agent systems. *IEEE Access*, 8:182537–182549.

Souza de Jesus, V., Mori Lazarin, N., Pantoja, C. E., Vaz Alves, G., Ramos Alves de Lima, G., and Viterbo, J. (2023). An ide to support the development of embedded multi-agent systems. In Mathieu, P., Dignum, F., Novais, P., and De la Prieta, F., editors, *Advances in Practical Applications of Agents, Multi-Agent Systems, and Cognitive Mimetics. The PAAMS Collection*, pages 346–358, Cham. Springer Nature Switzerland.

Wooldridge, M. (2009). *An Introduction to MultiAgent Systems*. Wiley Publishing, 2nd edition.