Controle de Tráfego Aéreo Autônomo: Simulação de Sistemas Multiagentes com o framework JaCaMo na plataforma Unity

Bernardo Viero[1], **Rafael H. Bordini**[2], **Alexandre Zamberlan**[1]

[1]Curso de Ciência da Computação – Universidade Franciscana (UFN)
Santa Maria – RS

[2]Escola Politécnica – PUCRS
Porto Alegre – RS

{bernardo.viero,alexz}@ufn.edu.br, rafael.bordini@pucrs.br

Abstract. *This research aims to unite the areas of Multi-Agent Systems (MAS) and Simulation in the management and control of autonomous air traffic. The simulated agents represent aircraft, control towers, and runways for landing and takeoff, with the ability to perceive, plan, execute, and communicate with each other. The behavior of the agents should be emergent and dynamic, meaning that the simulations should generate states or situations requiring the agents to plan, communicate, and negotiate. For this, the Java and AgentSpeak(L) languages, and the Jason interpreter are used for server construction, while the C# language and the Unity platform are used for the implementation of the simulations.*

1. Introdução

No universo complexo da gestão e controle do tráfego aéreo, a Simulação Computacional desempenha um papel crucial na tomada de decisões e na garantia da segurança das operações. A interação entre Sistemas Multiagentes e Simulação Computacional abre portas para abordagens desafiadoras na gestão do tráfego aéreo autônomo [1]. Ao unir essas áreas, é possível criar ambientes dinâmicos e emergentes, onde agentes simulados representam não apenas aeronaves, mas também torres de controle e pistas de pouso, dotados de capacidades de percepção, planejamento, execução e comunicação entre si. Isso se traduz em uma ferramenta valiosa para investigar cenários variados, avaliar estratégias de controle e mitigar riscos associados ao tráfego aéreo não tripulado. Além disso, ao utilizar tecnologias como Java, *framework* JaCaMo, AgentSpeak(L) e Unity, podem-se criar ambientes interativos e visualmente ricos, que facilitam o monitoramento e a análise do comportamento inteligente das aeronaves.

Neste contexto, este projeto propõe não apenas projetar e implementar um ambiente de simulação baseada na teoria de Sistemas Multiagente, mas também avaliá-lo de alguma forma. A integração de Jason em um ambiente de simulação Unity oferece a oportunidade de avaliar e aprimorar o desempenho do sistema em diferentes cenários, contribuindo assim para qualificar as discussões na área de controle de tráfego aéreo autônomo.

1.1. Controle de Tráfego Aéreo e Simulação

O controle de tráfego aéreo é um sistema essencial para garantir a segurança e a eficiência das operações aéreas em todo o mundo. Em sua essência, é responsável por gerenciar o

fluxo de aeronaves nos céus, coordenando suas rotas, altitudes e velocidades para evitar colisões e garantir o fluxo ordenado do tráfego aéreo [2]. O funcionamento do controle de tráfego aéreo envolve uma rede complexa de centros de controle, torres de controle e sistemas de comunicação. Nos centros de controle, controladores aéreos monitoram continuamente as posições das aeronaves por meio de radares e sistemas de navegação por satélite, como o GPS. Eles utilizam computadores e sistemas de gestão de tráfego aéreo para acompanhar o movimento das aeronaves e determinar as melhores rotas e altitudes para cada voo.

A simulação é um método que envolve a criação de um modelo de um sistema real e a condução de experimentos com esse modelo, com o objetivo de compreender os comportamentos do sistema ou avaliar estratégias para sua operação [3]. Essa prática pode ser vista como uma ferramenta computacional que permite o desenvolvimento, teste e estudo de teorias, visando a compreensão de sistemas reais e o desenvolvimento de sistemas computacionais [4]. Nesse contexto, os veículos aéreos são representados como agentes autônomos que possuem capacidade de percepção, planejamento, comunicação e tomada de decisão, com isso, são definidos como um Sistemas Sensíveis ao Contexto (*Context-Aware Computing*) [5].

1.2. Trabalhos Correlatos

Na pesquisa de Alexandre Zamberlan [6], foi conduzido um estudo que ressaltou a aplicação da teoria de Sistemas Multiagentes em Simulações Computacionais. Essas simulações foram realizadas em ambientes contendo partículas poliméricas nanoestruturadas, que consistiam em fármacos encapsulados em polímeros. No contexto, as partículas atuavam como agentes autônomos, interagindo entre si e com o ambiente. O trabalho empregou agentes e Sistemas Multiagentes com uma arquitetura reativa, visando monitorar se as partículas demonstravam comportamento de aglomeração ou não, influenciado pela carga elétrica de cada uma e pelo pH do ambiente. Todas as interações no sistema seguiram as regras de colisão da teoria Browniana do movimento.

No trabalho de Gabriel Dal Forno [7], foi ressaltado a implementação de um sistema multiagentes baseado em tráfego urbano autônomo. A arquitetura da simulação foi baseada em um simulador (ambiente Unity e linguagem C#) e um sistema multiagentes (ambiente Jason) com a utilização de *Swarm Intelligence* [8]. Com isso, o sistema multiagentes comportou-se como um servidor e o simulador como um cliente, comunicando-se via protocolo TCP/IP. A metodologia utilizada, para a implementação do sistema multiagentes, foi através do PROMETHEUS [9]. Já os resultados alcançados foram pertinentes, como o estabelecimento da comunicação eficaz entre o simulador e o sistema multiagentes através de *sockets* e o uso de *JavaScript Object Notation* (JSON), e a implementação dos planos que lidam com os eventos ativadores dos agentes em Jason.

O projeto de Carlos Pantoja [10] desenvolveu um agente inteligente para controlar uma aeronave em um simulador de voo, com foco na integração do Jason com o simulador de voo X-Plane. O agente foi responsável por todas as tarefas de voo, incluindo decolagem, navegação e pouso. O agente foi programado usando técnicas de aprendizado de máquina e controle de sistemas. Foi fixado um aeroporto para que o agente pudesse realizar a decolagem, e também foi necessário a criação de planos para que o agente pudesse decolar de qualquer aeroporto sem uma prévia configuração. O agente apresentou um

bom controle da aeronave na maioria das simulações, porém em algumas ocasiões houve desvios do percurso durante a decolagem e inclinações horizontais que não foram corrigidas. Esses comportamentos foram atribuídos à simplicidade do raciocínio do agente e às condições climáticas variáveis do simulador. No entanto, a codificação de planos e o acesso a informações adicionais permitiram melhorias significativas no controle da aeronave.

2. Metodologia

Na condução da pesquisa, adotou-se a metodologia Scrum, conforme descrita por Silveira [11], juntamente com a técnica Kanban para gerenciar as atividades assumidas. Os ciclos de trabalho, *sprints*, ocorrem semanalmente, baseando-se nas funcionalidades mapeadas e integradas ao Trello para Kanban. Para desenvolver o sistema multiagente (servidor da simulação), optou-se por utilizar o interpretador Jason, empregando as linguagens AgentSpeak(L) e Java, fundamentados na metodologia JaCaMo. Para o ambiente de simulação (cliente), foi selecionado o software Unity, utilizando C# como linguagem de programação. Esta escolha foi motivada pela variedade de classes disponíveis no Unity, que são úteis na criação de ambientes simulados, como colisores, vetores de movimento e facilidades para implementação de paralelismo. Por fim, foi decidido usar *sockets* para comunicação, por meio do protocolo TCP, entre a aplicação de simulação e a aplicação do sistema multiagentes, devido à sua fácil implementação em ambas as linguagens utilizadas.

Para a pesquisa, aplica-se um ambiente de avaliação e teste que trate das metas e planos dos agentes (torre de comando, pista, aviões); dos valores das variáveis para aviões (altura, nível de combustível e se o aeroporto é uma escala desse avião em voo); e da quantidade de aviões a decolar e/ou pousar. Com isso, busca-se avaliar se o auto gerenciamento dos agentes, por meio de negociação e planejamento descentralizado, minimiza as filas de espera e/ou promove um fluxo mais eficiente de decolagens e pousos. A avaliação do funcionamento dos agentes deve ser realizada por meio dos cenários definidos pelo usuário. Cada cenário é testado com diferentes valores para as variáveis mapeadas, que geram volumes de tráfego aéreo, visando entender o sistema.

Como a metodologia JaCaMo integra três plataformas: Jason, CArtAgO e Moise, há alguns passos a seguir:

Identificação dos Agentes e suas características:

- Agente Avião com propriedades como altitude, nível de combustível e se tem escala naquele aeroporto. Comportamentos como decolar, voar, pousar;
- Agente Pista com propriedade como fila de aviões prontos para decolar;
- Agente Torre de Controle com a responsabilidade de gerenciar o fluxo de aviões, tanto para decolagem quanto para pouso.

Definição de Artefatos (usados pelos agentes para interagir com o ambiente):

- Artefato Fila de Decolagem que gerencia a fila de aviões esperando para decolar;
- Artefato Fila de Pouso que gerencia a fila de aviões esperando para pousar.

Estrutura Organizacional (Moise):

- Papéis: controlador de pouso (gerenciado pela Torre de Controle); controlador de decolagem (gerenciado pela Torre de Controle); piloto (papel assumido pelos agentes avião);

- Normas: aviões devem negociar entre si e receber autorização para decolar ou pousar; a Torre de Controle deve garantir comunicação a todos os agentes.

Com base nas definições discutidas, o diagrama da Figura 1 fornece uma visão geral do sistema multiagente usando a metodologia JaCaMo, mostrando como os agentes interagem com os artefatos para realizar o controle descentralizado do tráfego aéreo. O diagrama da Figura 2, mostra o fluxo dos processos (modelagem de funcionamento do simulador) para a execução da simulação com relação ao usuário, ao simulador e ao Sistema Multiagente.

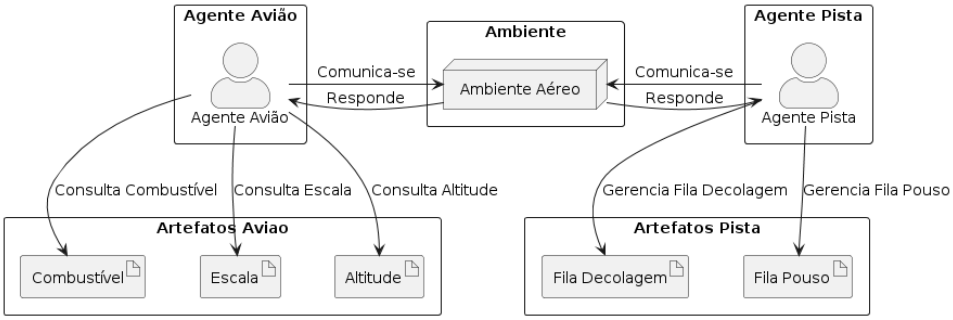

Figura 1. Diagrama de visão geral do sistema multiagente [12].

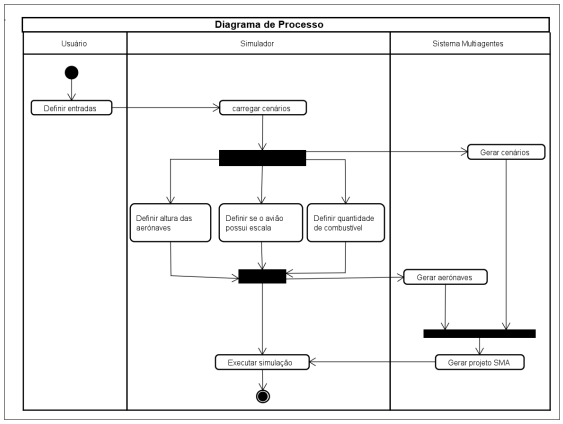

Figura 2. Diagrama do processo da simulação [12].

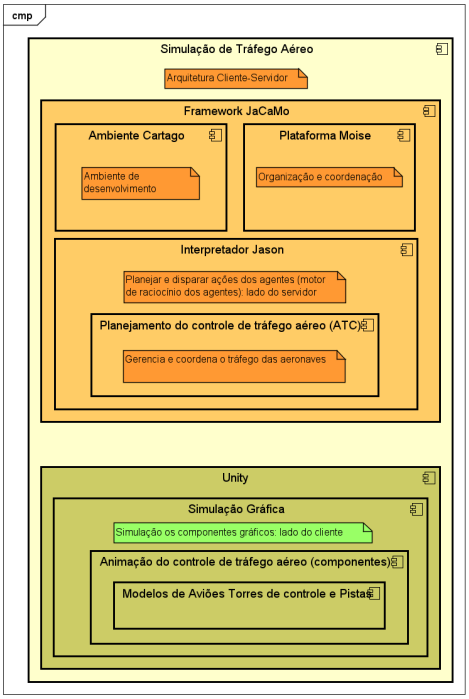

Figura 3. Diagrama de componentes [12].

As simulações gráficas, ao receberem comunicação do JaCaMo por meio da plataforma Unity, podem gerenciar graficamente esse ambiente. Isso inclui a exibição de aeronaves com rotas planejadas e detecção de colisões, torres de controle e pistas de pouso e decolagem. Conforme observado na Figura 3, a arquitetura adotada pelo projeto

é de natureza cliente-servidor. O servidor corresponde à implementação do *framework* JaCaMo, encarregado de acionar ações por meio do interpretador Jason para a linguagem AgentSpeak(L) conforme as circunstâncias de cada agente. Desse modo, o sistema de controle de tráfego aéreo é capaz de gerenciar e coordenar o fluxo das aeronaves. Por sua vez, o cliente assume a responsabilidade de receber tais ações provenientes do servidor e realizar simulações gráficas por meio da plataforma Unity. Em outras palavras, toda a parte de animações do tráfego aéreo, bem como os modelos de aeronaves e pistas, é renderizada por essa plataforma.

Decidiu-se o uso de *socket* para comunicação, via protocolo TCP (tanto do servidor ao simulador, quanto do simulador ao servidor), entre a aplicação de simulação e a aplicação do Sistema Multiagente, devido à sua implementação fácil em ambas as linguagens empregadas, além de interconectar dois sistemas heterogêneos (servidor em Java, com AgentSpeak(L) e Jason; cliente em C#, com Unity). A Figura 4 ilustra a dinâmica entre o simulador e o Sistema Multiagente em relação à comunicação entre as tecnologias. A Figura 5 apresenta uma proposta inicial de interface gráfica para o sistema, destacando a visualização da simulação.

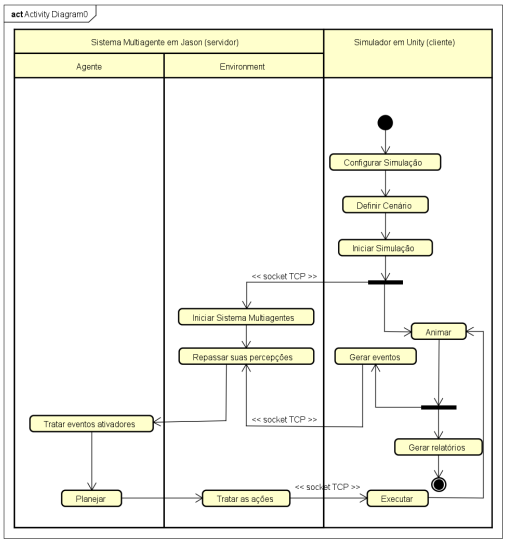

Figura 4. Diagrama de atividades para integração do simulador [12].

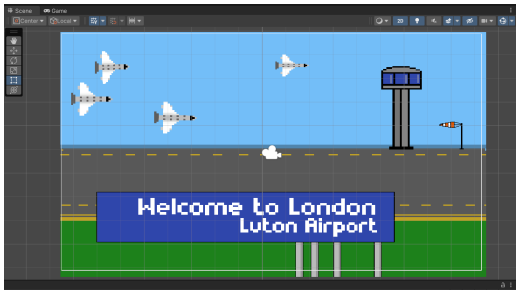

Figura 5. Protótipo de simulação [12].

Com o uso do sistema de comunicação baseado em *socket*, onde o Sistema Multiagente abriga o servidor e o ambiente de simulação abrange o cliente, a troca de informações entre os dois lados é realizada por meio de *strings* no formato JSON. Esse formato foi escolhido devido à disponibilidade de bibliotecas de codificação e decodificação em ambas as linguagens utilizadas. As informações enviadas do cliente para o servidor contêm dados relativos aos agentes e outros objetos. Enquanto as instruções destinadas às aeronaves da simulação, definidas pelos agentes do Sistema Multiagente, são transmitidas do servidor para o cliente. O protocolo TCP foi selecionado para a implementação, pois a perda de uma instrução/ação do Sistema Multiagente poderia causar falta de sincronia entre os dois lados e resultar em erros que poderiam levar a colisões ou acidentes na simulação.

3. Considerações finais

Este trabalho propõe um projeto de um sistema de simulação de tráfego aéreo por meio de sistemas multiagentes. Especificamente, o sistema modelado consiste de aviões em um aeroporto e uma torre de controle que devem ser capazes de se comunicar e auto-organizar, visando a geração de um fluxo de tráfego eficiente. Além disso, os agentes do sistema devem aprender a interagir com outros agentes independentes.

A pesquisa encontra-se na fase final de modelagem e especificação do sistema de simulação via SMA. Toda a funcionalidade de comunicação entre o servidor (Jason) e o ambiente de simulação (Unity) já está preparada, aguardando a implementação dos comportamentos dos agentes e dos cenários de simulação.

Referências

[1] Wenhui Fan et al. "Multi-Agent Modeling and Simulation in the AI Age". Em: *TSINGHUA SCIENCE AND TECHNOLOGY* (2021). URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9409754.

[2] Elmira Zohrevandi et al. "Design and evaluation study of visual analytics decision support tools in air traffic control". Em: *Computer Graphics Forum*. Vol. 41. 1. Wiley Online Library. 2022, pp. 230–242.

[3] "Como construir modelos de simulação válidos e confiáveis". Em: *Conferência de Simulação de Inverno 2022 (WSC)*. IEEE, 2022, pp. 1283–1295.

[4] Adelinde M. Uhrmacher e Danny Weyns. *Multi-Agent Systems: Simulation and application*. Computational analysis, synthesis, and design of dynamic models series. Boca Raton, FL, USA: CRC Press, 2009.

[5] Mir Salim Ul Islam, Ashok Kumar e Yu-Chen Hu. "Context-aware scheduling in Fog computing: A survey, taxonomy, challenges and future directions". Em: *Journal of Network and Computer Applications* 180 (2021), p. 103008.

[6] Alexandre Zamberlan et al. "Multi-Agent Systems, Simulation and Nanotechnology". Em: *Multi Agent Systems-Strategies and Applications*. IntechOpen, 2020.

[7] Gabriel Dal Forno, Rafael Heitor Bordini e Alexandre Zamberlan. *Simulação de tráfego autônomo com Inteligência Coletiva e Emergente*. Pimenta Cultural - Upgrade: jogos, entretenimento e cultura, 2024. URL: https://www.pimentacultural.com/livro/upgrade-jogos-2/ (acesso em 20/05/2024).

[8] Eric Bonabeau, Marco Dorigo e Guy Theraulaz. *From Natural to Artificial Swarm Intelligence*. USA: Oxford University Press, Inc., 1999. ISBN: 0195131584.

[9] Lin Padgham e Michael Winikoff. *Developing Intelligent Agent Systems: A practical guide.* John Wiley & Sons, 2004.

[10] Carlos Eduardo Pantoja e Tielle da Silva Alexandre. "Um Agente Inteligente para Simulação de Voo Usando Jason e X-Plane". Em: *8th Software Agents, Environments and Applications School (WESAAC)* (2014).

[11] Paulo Silveira et al. *Introdução à Arquitetura de Design de Software*. Ed. por Elsevier. 2011.

[12] Bernardo Viero. *images-tfg*. 2024. URL: https://github.com/bernardoviero/TFG/tree/main/images-tfg (acesso em 18/06/2024).