Proposta de uma extensão do VSCode para facilitar a adoção do JaCaMo no desenvolvimento de Sistemas Multiagente

Mustafa Ribeiro de Almeida Neto, Nilson Lazarin, Bruno Freitas, Diego Castro

Bacharelado em Sistemas de Informação – Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (Cefet/RJ) – Nova Friburgo, RJ – Brazil

chon@grupo.cefet-rj.br

Abstract. *The development of systems using the Multi-Agent Oriented Programming (MAOP) paradigm presents challenges for young developers, due to the combination of several tools, the need to use the command line, and the long learning curve. Although some solutions exist, they aim to facilitate the adoption of this paradigm, but they are still depreciated or need improvements. This paper aims to present an extension of the JaCaMo framework for the VSCode development environment, integrating functions such as project initialization, code execution, and syntax highlighting, reducing technical obstacles, and improving the programmer's work process.*

Resumo. *O desenvolvimento de sistemas usando o paradigma de programação orientada a multiagentes apresenta desafios para programadores iniciantes, devido à combinação de várias ferramentas, à necessidade de uso de linha de comando e à longa curva de aprendizado. Apesar de existirem, algumas soluções que buscam facilitar a adoção deste paradigma, se encontram depreciadas ou ainda precisam de aprimoramentos. Diante disso, o presente trabalho apresenta uma extensão do framework JaCaMo para o ambiente de desenvolvimento VSCode, integrando funções como a inicialização de projetos, execução de código e destacação na sintaxe, diminuindo obstáculos técnicos e melhorando o processo de trabalho do programador.*

1. Introdução

Os Sistemas Multiagentes (SMA) destacam-se como uma abordagem para a modelagem e implementação de sistemas distribuídos, compostos por múltiplos agentes autônomos, capazes de perceber o ambiente, se adaptar ao contexto, interagir com outros agentes e tomar decisões deliberativas de forma independente [Bordini et al. 2020]. Para apoiar o desenvolvimento de sistemas baseados em agentes, a Programação Orientada a Multiagentes (*Multi-Agent Oriented Programming* – MAOP) propõe um modelo de engenharia baseado em três dimensões fundamentais: agente, ambiente e organização [Boissier et al. 2013]. Nesse contexto, o framework JaCaMo destaca-se como a concretização do MAOP, oferecendo uma infraestrutura completa para o desenvolvimento de SMA. Entretanto, apesar de ser uma solução robusta, sua adoção, para desenvolvedores iniciantes, é custosa. A criação de projetos requer o uso de ferramentas de linha de comando e a configuração de múltiplos arquivos com sintaxes distintas — como *.jcm*, *.asl*, *.java* e *.xml* — organizados segundo as três dimensões da arquitetura MAOP.

No contexto da facilitação da adoção do MAOP para iniciantes, duas iniciativas se destacam. A primeira é o *JaCaMo-Web* [Amaral and Hübner 2020], que propõe

uma abordagem de programação interativa, permitindo criar, inspecionar, modificar e destruir agentes, artefatos e organizações em tempo de execução, sem necessidade de reinicialização do sistema. Essa funcionalidade reduz o tempo de desenvolvimento e facilita a prototipação incremental, oferecendo uma interface web acessível baseada em uma API RESTful. A segunda é a *ChonIDE* [Souza De Jesus et al. 2023], uma IDE voltada ao desenvolvimento de Sistemas Multiagentes Embarcados, que integra funcionalidades tanto da camada de raciocínio (agentes) quanto da camada de firmware (microcontroladores). Sua proposta é centralizar, em um único ambiente, os principais recursos necessários à embarcação, execução e monitoramento de SMA, permitindo inclusive o acesso remoto à interface embarcada em um dispositivo físico.

Apesar dessas contribuições relevantes, o *JaCaMo-Web* não substitui um ambiente completo de desenvolvimento, pois depende de ferramentas como `gradle`, `docker` e `node.js`, exigindo familiaridade com a configuração de projetos via terminal. Além disso, seu foco está na manipulação dinâmica de instâncias em tempo de execução, sem oferecer suporte à modelagem e organização do ciclo de vida completo dos projetos. A criação e configuração inicial precisam ser realizadas manualmente, fora da interface. De forma semelhante, embora represente um avanço importante, especialmente para aplicações distribuídas e embarcadas, a *ChonIDE* apresenta limitações significativas. Seu escopo é restrito a dimensão dos agentes, sem suporte às demais dimensões da arquitetura JaCaMo. Sua compatibilidade é limitada a sistemas baseados em Linux, o que dificulta sua adoção em contextos educacionais diversos, como laboratórios com computadores Windows. Além disso, a *ChonIDE* não atende plenamente aos critérios de usabilidade esperados de IDEs modernas, como instalação multiplataforma, checagem de sintaxe, funções para autocompletar, múltiplas abas e interface intuitiva [Siqueira et al. 2024]. Diante dessas limitações, torna-se evidente a necessidade de ferramentas que abstraiam a complexidade técnica envolvida no uso do JaCaMo, proporcionando um ambiente de desenvolvimento mais acessível, integrado e voltado à aprendizagem e à prototipagem.

Este trabalho propõe o desenvolvimento de uma extensão para o Visual Studio Code (VSCode), com o objetivo de simplificar a criação e execução de projetos JaCaMo. Fornecendo uma interface gráfica intuitiva, eliminando a necessidade de interação direta com o terminal para operações básicas, como criação de arquivos, estruturação de diretórios e execução de projetos. Dessa forma, busca-se melhorar a experiência do usuário e ampliar o acesso à plataforma JaCaMo, especialmente entre iniciantes e em ambientes educacionais com computadores baseados no sistema operacional Microsoft Windows.

2. Background

O desenvolvimento de sistemas multiagentes envolve um conjunto de atividades fundamentais, como análise de requisitos, modelagem das interações, codificação dos comportamentos dos agentes, testes e manutenção. A natureza distribuída, autônoma e cooperativa desses sistemas impõe desafios adicionais ao processo de desenvolvimento, exigindo não apenas o domínio de conceitos como percepção, raciocínio e comunicação entre agentes, mas também o uso de ferramentas que ofereçam suporte a essas abstrações. Estudos recentes [Souza De Jesus et al. 2023, Siqueira et al. 2024, Lima et al. 2025] destacam que a ausência de ambientes de desenvolvimento adequados pode comprometer a produtividade, dificultar o aprendizado de novos desenvolvedores e restringir a adoção de soluções baseadas em agentes.

Nesse contexto, os *Integrated Development Environments* – IDEs desempenham um papel essencial ao consolidar funcionalidades como edição de código, execução, depuração e gerenciamento de projetos em um único ambiente. Essa integração facilita a codificação, contribui para a automação de tarefas, melhora a identificação de erros e reduz a incidência de falhas, promovendo a produtividade e a qualidade do software gerado. Dentre as IDEs amplamente utilizadas, destaca-se o *Visual Studio Code* (VSCode), conhecido por sua flexibilidade e alto grau de extensibilidade. O VSCode permite a personalização de seu ambiente por meio de extensões, que adicionam funcionalidades específicas como preenchimento automático inteligente, realce de sintaxe, execução de comandos automatizados e suporte a múltiplas linguagens de programação.

O diferencial arquitetural do VSCode está no paradigma *extension-in-IDE*, no qual extensões funcionam como miniaplicações incorporadas ao núcleo da IDE. Esse modelo permite que funcionalidades especializadas sejam integradas de maneira nativa, leve e não intrusiva, utilizando elementos visuais e operacionais do próprio editor. Essa abordagem favorece a adaptação da ferramenta a diferentes domínios e fluxos de trabalho, sem a necessidade de adoção de um ambiente completamente novo por parte do desenvolvedor [Liu et al. 2025].

Recentes trabalhos têm explorado com sucesso esse paradigma para a criação de extensões, como por exemplo: i) a *mn-analise* que integra o ChatGPT ao VSCode para auxiliar na detecção de vulnerabilidades de segurança em código-fonte [Almeida Neto and Lazarin 2024]; ii) a *VSCode Migrate*, voltada à realização de migrações semi-automáticas em projetos com baixa cobertura de testes, explorando a criação de interfaces customizadas e edição orientada por scripts externos [Vahlbrock et al. 2022]; iii) a *Divinator*, uma extensão baseada em aprendizado profundo para geração de resumos automáticos em linguagem natural a partir de trechos de código em Java, Python e C#, utilizando uma arquitetura de microsserviços [Durelli et al. 2022]. Esses exemplos demonstram como o paradigma *extension-in-IDE* tem sido explorado para oferecer suporte a funcionalidades avançadas — como segurança, refatoração e compreensão automática de código — mantendo a usabilidade e familiaridade do ambiente de desenvolvimento. Tal versatilidade reforça a adequação do VSCode como plataforma-base para o desenvolvimento de soluções especializadas, como a proposta deste trabalho voltada ao suporte ao framework JaCaMo.

Além disso, foi realizada uma busca na loja oficial de extensões do Visual Studio Code, onde foram identificadas apenas quatro extensões relacionadas ao JaCaMo, conforme apresentado na Tabela 1. Todas essas ferramentas oferecem apenas suporte básico de realce de sintaxe para arquivos específicos do ecossistema JaCaMo, como *.asl*, *.mas2j* e *.jcm*, não disponibilizando funcionalidades mais avançadas como criação automatizada de projetos, execução interativa ou integração entre as dimensões de agente,

Tabela 1. Extensões do VSCode para ao JaCaMo disponíveis na loja oficial.

Extensão	Cobertura funcional	Atualização	Downloads
JaCaMo4Code	Realce de sintaxe e snippets para arquivos JaCaMo (`.asl` e `.jcm`)	ago/2020	6.941
code-mas2j	Realce de sintaxe para arquivos `.asl` e `.mas2j`	mai/2018	1.786
AgentSpeak	Realce de sintaxe para arquivos `.asl`	jul/2021	475
VS Code JaCaMo	Realce de sintaxe para arquivos `.jcm`	mai/2019	542

ambiente e organização. Ademais, a maioria dessas extensões encontra-se desatualizada, com seus últimos commits realizados entre 2018 e 2021, o que reforça a ausência de uma solução mantida e abrangente para apoiar o desenvolvimento de SMAs com JaCaMo através do VSCode. Isso fortalece a justificativa para a proposta deste trabalho, que visa justamente suprir essa lacuna por meio de uma extensão moderna, funcional e alinhada às boas práticas de usabilidade e integração do paradigma *extension-in-IDE*.

3. Proposta

O processo tradicional para desenvolvimento de aplicações com JaCaMo, exige que o desenvolvedor realize manualmente a instalação do Java e a execução de múltiplos comandos Gradle em um terminal. Ferramentas auxiliares, como o Jacamo-CLI, foram criadas para simplificar este ecossistema, mas introduzem seus próprios desafios: a sua instalação demanda que o usuário clone o repositório do GitHub, compile a ferramenta a partir do código-fonte com o comando ./gradlew createBin e mova o binário manualmente para um diretório de sistema, como /usr/bin. Crucialmente, este script de compilação não é compatível com sistemas Windows (cenário verificado em junho de 2025), o que limita drasticamente a adoção da ferramenta por uma grande parcela de desenvolvedores.

Ao centralizar essas funcionalidades em uma única extensão e aproveitando a infraestrutura consolidada da IDE, esta proposta busca facilitar o aprendizado, a experimentação e a adoção do JaCaMo, atendendo tanto a iniciantes quanto a desenvolvedores experientes. A extensão busca integrar ferramentas essenciais diretamente no editor, proporcionando uma experiência de desenvolvimento mais fluida e eficiente. Ela adota o paradigma *extension-in-IDE*, integrando recursos específicos diretamente à interface do VSCode, interagindo diretamente com o jacamo-cli, que por sua vez utiliza-se do gradle, que compila o projeto usando o java, conforme Figura 1.

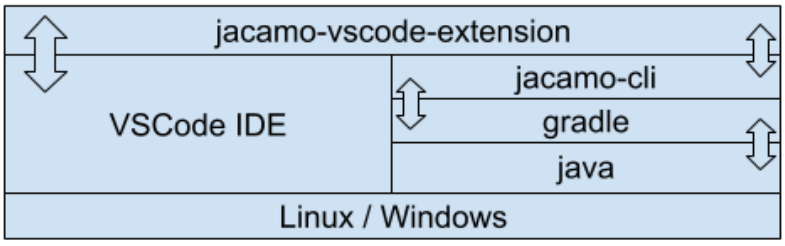

Figura 1. Interação da extensão com outros componentes.

A seguir, são apresentadas a arquitetura da extensão e os fluxos de interação com o usuário. As funcionalidades previstas têm como objetivo: i) **Facilitar a criação de projetos**, automatizando a geração da estrutura de diretórios e arquivos compatíveis com o paradigma MAOP; ii) **Simplificar a execução de SMA**, oferecendo controle gráfico para compilar, iniciar e parar projetos diretamente na interface do VSCode; iii) **Apoiar a codificação multidimensional**, fornecendo recursos específicos para edição de arquivos relacionados às linguagens Jason, CArtAgO e Moise+; iv) **Garantir compatibilidade multiplataforma**, promovendo o funcionamento fluido da extensão em sistemas Windows e Linux. A Tabela 2 apresenta um resumo da implementação técnica de cada uma dessas funcionalidades.

Tabela 2. Implementação das funcionalidades propostas para a extensão.

Funcionalidade	Implementação
Criação de Projetos	Botão na interface que executa o a criação de projetos via `jacamo-cli`.
Execução Interativa	Botões na interface acionam comandos do `jacamo-cli`, com exibição de logs no terminal integrado.
Suporte à Codificação	*Syntax highlighting* para arquivos `.asl` (Jason), `.java` (CArtAgO) e `.jcm` (Moise+).
Multiplataforma	Abstração de comandos do sistema operacional, garantindo compatibilidade com Windows e Linux.

4. Desenvolvimento

Esta seção detalha a metodologia adotada, incluindo as tecnologias empregadas, a configuração do ambiente de desenvolvimento e os critérios de avaliação da ferramenta. A primeira etapa consistiu na definição dos requisitos da extensão. Foram realizadas algumas entrevistas com usuários JaCaMo, participantes de grupos de pesquisa sobre agentes, para entender as funcionalidades desejadas. Entre os requisitos coletados, a disponibilização dos comandos do JaCaMo por meio de uma *Activity Bar*, fornecendo botões de fácil acesso, teve destaque, demonstrando dificuldade de utilização de linhas de comandos por usuários iniciantes. Além disso, requisitos para o suporte a realce de sintaxe para arquivos .jcm e publicação da aplicação do *Marketplace* para garantir fácil instalação e atualização da extensão também foram coletados.

O desenvolvimento seguiu as diretrizes para extensões do VSCode, garantindo compatibilidade e boas práticas de desenvolvimento. Para a implementação da extensão, o Node.js foi utilizado, aproveitando seu robusto ecossistema para desenvolvimento web, o que facilitou a integração com a API do VSCode, responsável pela interação direta com o editor, permitindo a criação de comandos, configurações personalizadas e integração com a interface do usuário. Para o empacotamento, publicação e testes, foi necessária a utilização do VSCode Extension CLI (VSCE) que permitiu a fácil execução da ferramenta de forma nativa pela IDE. Por fim, todo o projeto foi desenvolvido seguindo boas práticas de versionamento de código, sendo apoiado pelo rastreamento de *issues* no GitHub.

Além das tecnologias citadas acima, o desenvolvimento da extensão envolveu a utilização de dependências específicas para testes, análise de código e compatibilidade com o VSCode. Para análise, foram utilizadas as bibliotecas de tipagem do TypeScript, como @types/vscode, @types/mocha e @types/node, o que permitiu uma boa padronização do código, sendo apoiado ainda pelo eslint, o que permitiu uma padronização em relação às práticas de escrita de código, como indentação, quebra de linha, agrupamento de importações, etc. Por fim, as bibliotecas @vscode/test-cli e @vscode/test-electro ficaram encarregadas de executar os testes automatizados da aplicação criada.

4.1. Facilitadores da distribuição do JaCaMo-CLI para Windows e Linux

Como evidenciado anteriormente, o JaCaMo-CLI precisa ser compilado manualmente, porém, para usuários Windows, o script de compilação não está disponível no site do projeto. Dessa forma, realizadas algumas ações para facilitar a distribuição do JaCaMo. Para sistemas Linux derivados do Debian, o repositório do projeto do JaCaMo-CLI foi baixado e compilado (conforme instruções para geração do binário) e posteriormente empacotado para o gerenciador de pacotes APT, com dependências necessárias já definidas (*openjdk-21-jdk ou temurin-21-jdk*) de forma que, durante a instalação, as dependências possam

ser resolvidas e instaladas automaticamente. Posteriormente, o pacote foi hospedado no servidor público de pacotes do grupo de pesquisa.

Para sistemas Windows, foi necessário baixar o projeto e, através de uma IDE para aplicações Java (IntelliJ), foi gerado um artefato (arquivo .jar). Porém, para facilitar a execução no Windows, foi necessário transformar o artefato em um executável (arquivo .exe). Para isso, foi utilizado o Launch4j, uma ferramenta multiplataforma para encapsular aplicações Java distribuídas como jars em executáveis nativos do Windows. Por fim, foi necessário criar um instalador para o sistema que incluísse as dependências JAVA, evitando assim duas instalações. Para a construção do instalador foi utilizado o Inno Settup, um instalador gratuito para programas do Windows criado por Jordan Russell e Martijn Laan. No projeto do instalador, incluiu o JDK 21 distribuído pelo projeto Eclipse Temurin, fornecedor do OpenJDK (a implementação de referência do Java) gratuito e de código aberto. Após a compilação do instalador para Windows, ele também foi hospedado no servidor público do grupo de pesquisa.

4.2. Desenvolvimento da Extensão

O desenvolvimento da extensão seguiu um fluxo estruturado, dividido em etapas que cobrem desde a configuração inicial até a publicação no Marketplace. A Figura 2 apresenta uma visão do fluxo de funcionamento da extensão, desde a ativação de comandos pelo usuário até a execução por meio da JaCaMo CLI e a exibição dos resultados no terminal do VSCode. Essa visão oferece um panorama geral do processo, conectando os principais arquivos da extensão com as ações no ambiente de desenvolvimento.

A estrutura do projeto foi criada utilizando a ferramenta VSCE (Visual Studio Code Extension), que fornece os arquivos base para a construção de extensões no VSCode. Em seguida, o arquivo package.json foi configurado para registrar comandos, permissões e atalhos, definindo o comportamento da extensão no ambiente do editor.

A extensão foi desenvolvida com o objetivo de simplificar o ciclo de vida de aplicações baseadas no framework JaCaMo. Para isso, foram implementadas as seguintes funcionalidades principais: **Criação de Aplicações JaCaMo:** Um comando permite gerar automaticamente a estrutura inicial de um projeto JaCaMo, requisitando apenas o nome da aplicação ao usuário. **Execução de Aplicações:** O usuário pode executar arquivos .jcm diretamente pelo VSCode, sem necessidade de ferramentas externas. A extensão detecta e gerencia múltiplas execuções simultâneas, evitando conflitos. **Parada de Execução (MAS):** Comando que identifica e finaliza agentes em execução no ambiente atual, utilizando comandos do jacamo-cli. **Painel de Saída (Output Panel):** Todos os logs são exibidos em um painel integrado no VSCode. Os dados são formatados com

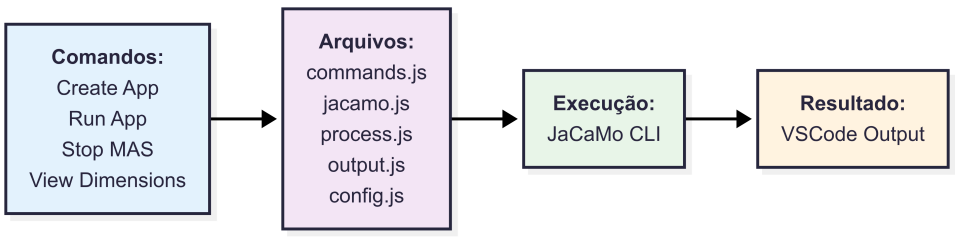

Figura 2. Fluxo geral da extensão: comandos, arquivos, execução e saída.

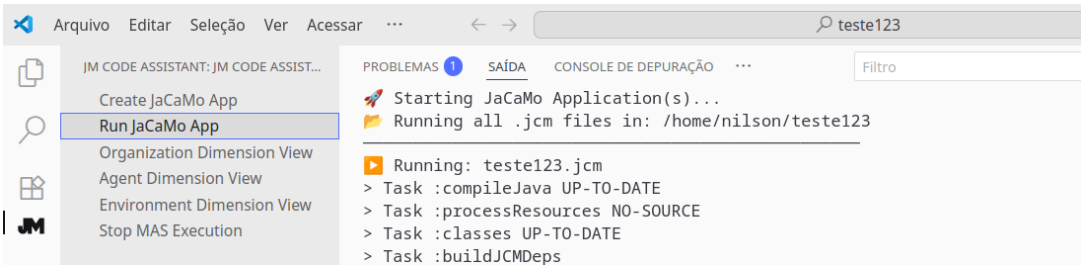

Figura 3. Ícone da extensão no Activity Bar com botões interativos.

ícones e destaques visuais, facilitando a leitura e a depuração. **Acesso às Interfaces Web:** A extensão oferece atalhos diretos para abrir as interfaces gráficas de monitoramento das dimensões organizacional, de agentes e de ambiente.

Para melhorar a usabilidade, a extensão inclui uma interface visual no painel lateral (Activity Bar) do VSCode. Nela, o usuário encontra botões interativos para executar as funcionalidades descritas, sem a necessidade de utilizar linha de comando ou menus.

A extensão oferece suporte ao realce de sintaxe para arquivos por meio da reutilização dos arquivos da extensão JaCaMo4Code [Krausburg 2020]. Essa integração permite uma experiência de codificação mais fluida e contextualizada.

Concluído o desenvolvimento, a extensão foi empacotada com o comando vsce package, gerando o arquivo .vsix para distribuição. A publicação foi realizada no Marketplace oficial do VSCode, tornando a ferramenta acessível a outros desenvolvedores.

4.3. Reprodutibilidade

Buscando garantir a reprodutibilidade, uma página[1] está disponível com os fontes da extensão e os facilitadores de instalação além de vídeos com instruções para execução da extensão.

5. Conclusão

Os testes realizados demonstraram que a extensão cumpre seu objetivo de facilitar o desenvolvimento de projetos JaCaMo no Visual Studio Code. A funcionalidade de criação de projetos permite a geração automática de toda a estrutura necessária, eliminando a necessidade de configuração manual. A simplicidade e eficácia da extensão tornam-na uma ferramenta útil para desenvolvedores que trabalham com JaCaMo, independentemente do nível de experiência. Para usuários iniciantes, a interface intuitiva reduz a curva de aprendizado, enquanto para usuários mais experientes, a automação de tarefas repetitivas melhora a produtividade. Os resultados indicam que a extensão pode contribuir para a adoção e o uso mais eficiente do JaCaMo no desenvolvimento de sistemas multiagentes.

Para aprimorar ainda mais a extensão e torná-la acessível a um público mais amplo, algumas melhorias futuras são sugeridas: i) Expansão do suporte ao debugging: implementação de ferramentas que permitam aos desenvolvedores depurar agentes JaCaMo diretamente no VSCode, facilitando a identificação e resolução de erros. ii) Melhoria na interface de usuário: aprimoramentos na interface visual para tornar a utilização da

[1] https://papers.chon.group/WESAAC/2025/jacamo-vscode-extension/

extensão ainda mais intuitiva e acessível. Além dessas, também é pretendida a execução de avaliações da extensão criada por meio da utilização de questionários de aceitação de novas tecnologias, como por exemplo o uso do questionário *Technology Acceptance Model* (TAM). Por meio dessa avaliação, é esperado dividir os participantes em dois grupos, um fazendo uso da ferramenta e outro sem a mesma, com o objetivo de demonstrar a facilidade de uso e comparar dificuldades.

Referências

Almeida Neto, M. R. d. and Lazarin, N. M. (2024). Uma extensão para o VSCode que utiliza o ChatGPT como ferramenta de apoio ao desenvolvimento de software seguro. In *COTB'24*, Balneário Camboriú. UNIVALI. DOI: 10.14210/cotb.v15.p310-311.

Amaral, C. J. and Hübner, J. F. (2020). Jacamo-Web is on the Fly: An Interactive Multi-Agent System IDE. In *Engineering Multi-Agent Systems*, volume 12058, pages 246–255. Springer International Publishing, Cham. DOI: 10.1007/978-3-030-51417-4_13.

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. DOI: 10.1016/j.scico.2011.10.004.

Bordini, R. H., El Fallah Seghrouchni, A., Hindriks, K., Logan, B., and Ricci, A. (2020). Agent programming in the cognitive era. *Autonomous Agents and Multi-Agent Systems*, 34(2):37. DOI: 10.1007/s10458-020-09453-y.

Durelli, R. S., Durelli, V. H. S., Bettio, R. W., Dias, D. R. C., and Goldman, A. (2022). Divinator: A Visual Studio Code Extension to Source Code Summarization. In *VEM 2022*, pages 1–5, Brasil. SBC. DOI: 10.5753/vem.2022.226187.

Krausburg, T. (2020). JaCaMo4Code - Visual Studio Marketplace. Disponível em: https://marketplace.visualstudio.com/items?itemName=tabajara-krausburg.jacamo4code.

Lima, G., Siqueira, E., Medeiros, T., Pantoja, C., Lazarin, N., and Viterbo, J. (2025). A Modularized and Reusable Architecture for an Embedded MAS IDE:. In *ICEIS 2025*, pages 1034–1045, Porto, Portugal. SCITEPRESS. DOI: 10.5220/0013439600003929.

Liu, Y., Tantithamthavorn, C., and Li, L. (2025). Protect Your Secrets: Understanding and Measuring Data Exposure in VSCode Extensions. In *IEEE SANER 2025*, pages 551–562, Montreal, QC, Canada. IEEE. DOI: 10.1109/SANER64311.2025.00058.

Siqueira, E., Ramos, G., Raboni, T., Pantoja, C., and Lazarin, N. (2024). Comparative Analysis of an IDE Prototype for Developing Embedded MAS. In *WESAAC 2024*, pages 85–95, Porto Alegre, RS, Brasil. SBC. DOI: 10.5753/wesaac.2024.33458.

Souza De Jesus, V., Lazarin, N. M., Pantoja, C. E., Vaz Alves, G., Lima, G. R. A. D., and Viterbo, J. (2023). An IDE to Support the Development of Embedded Multi-Agent Systems. In *PAAMS 2023*, Cham. Springer. DOI: 10.1007/978-3-031-37616-0_29.

Vahlbrock, T., Guddat, M., and Vierjahn, T. (2022). VSCode Migrate: Semi-Automatic Migrations for Low Coverage Projects. In *ICSME 2022*, Limassol, Cyprus. IEEE. DOI: 10.1109/ICSME55016.2022.00070.