REVISTA ELETRÔNICA CIENTÍFICA DA UERGS

Funcionalidades das plataformas de sistema multiagentes JADE e PADE em aplicações de manufatura industrial

João Alvarez Peixoto
Universidade Estadual do Rio Grande do Sul (UERGS).
E-mail: joao-peixoto@uergs.edu.br, http://lattes.cnpq.br/3242194031865969

André Borin Soares
Universidade Estadual do Rio Grande do Sul (UERGS).
E-mail: andre-soares@uergs.edu.br, https://lattes.cnpq.br/3591334903964533

Ramon Simeone Fagundes
Universidade Estadual do Rio Grande do Sul (UERGS).
E-mail: ramon-fagundes@uergs.edu.br, https://lattes.cnpq.br/4917070782354904

ISSN 2448-0479. Submetido em: 24 set. 2020. Aceito: 23 nov. 2021.
DOI: http://dx.doi.org/10.21674/2448-0479.81.32-42

Resumo

O uso de sistemas multiagentes vem surgindo como uma solução para proporcionar o autogerenciamento no fluxo produtivo da indústria, sem a necessidade da interferência humana. Atualmente, a alternativa funcional melhor documentada para esta finalidade é a plataforma JADE *(Java Agent DEvelopment Framework)*, o que implica no uso da linguagem de programação Java. Este trabalho tem como objetivo investigar uma nova alternativa a esta plataforma, utilizando a plataforma PADE *(Python Agent DEvelopment Framework)* para programar um sistema multiagentes dedicado à manufatura industrial. A partir de uma planta industrial simulada, foram executadas simulações com o uso deste sistema para o controle das máquinas usadas na produção de um produto que necessitava de uma montagem multiprocessos. Os resultados apresentaram que a plataforma PADE, mesmo sendo mais nova que a já consolidada JADE, demonstra ser um *framework* promissor, impulsionada pelo uso da linguagem de programação Python, em ascensão no meio computacional. Ter os resultados comparativos entre as duas plataformas em métricas distintas agrega valor quando da escolha de quais sistemas e qual linguagem utilizar a partir das métricas que o usuário necessitar.

Palavras-chave: Sistema multiagente; PADE; JADE; indústria 4.0; sistema auto-organizável.

Abstract

Features of the JADE and PADE multi-agent system platforms in industrial manufacturing applications.

The use of multi-agent systems has emerged as a solution to provide self-management in the industry's production flow, without the need for human interference. Currently, the best documented functional alternative for this purpose is the JADE platform (Java Agent DEvelopment Framework), which implies the use of the Java programming language. This work aims to investigate a new alternative to this platform, using the PADE platform (Python Agent DEvelopment Framework) to program a multi-agent system dedicated to industrial manufacturing. From a simulated industrial plant, simulations were performed with the use of this system to control the machines used in the manufacturing of a product, which required a multi-process assembly. The results showed the PADE platform, even though it is newer than the one already consolidated that JADE, proves to be a promising framework, driven by the use of the Python programming language, on the rise in the computing environment. Having the comparative results between the two platforms in different metrics adds value when choosing which systems and which language to use, based on the metrics that the user needs.

Keywords: Multiagent system; PADE; JADE; industry 4.0; self-organizing system.

Resumen

Características de las plataformas de sistemas multiagente JADE y PADE en aplicaciones de fabricación industrial

El uso de sistemas multiagente ha surgido como una solución para proporcionar la autogestión en el flujo de producción de la industria, sin la necesidad de interferencia humana. Actualmente, la alternativa funcional mejor documentada para este propósito es la plataforma JADE *(Java Agent DEvelopment Framework)*, que implica el uso del lenguaje de programación Java. Este trabajo tiene como objetivo investigar una nueva alternativa a esta plataforma, utilizando la plataforma PADE *(Python Agent DEvelopment Framework)* para programar un sistema multiagente dedicado a la fabricación industrial. A partir de una planta industrial simulada, se realizaron simulaciones con el uso de este sistema para el control de las máquinas utilizadas en la producción de un producto, que requería un ensamblaje multiprocesos. Los resultados evidenciaron que la plataforma PADE, aunque es más nueva que la ya consolidada que JADE, demuestra ser un *framework* promisor, impulsado por el uso del lenguaje de programación Python, en auge en el entorno computacional. Tener los resultados comparativos entre las dos plataformas en métricas diferentes agrega valor a la hora de elegir qué sistemas y qué lenguaje utilizar, en base a las métricas que el usuario necesitar.

Palabras clave: Sistema multiagente; PADE; JADE; industria 4.0; sistema de autoorganización.

Introdução

Na manufatura industrial, as fábricas produzem bens de consumo conforme a demanda dos consumidores; entretanto, como aponta Kotler, Kartajaya e Setiawan (2017), os consumidores têm exigido produtos cada vez mais diversificados. A flexibilidade nos sistemas produtivos (PÉREZ; BEDIA; FERNÁNDEZ, 2016) é o fator que leva este sistema a ser capaz de se modificar, seja estruturalmente ou em sua sequência de operação, de forma a mudar a regra do fluxo produtivo já estabelecido. Um sistema produtivo segue uma regra de fluxo de produção definida, mas deve ser flexível o suficiente para modificar esta regra em função de ocorrências no processo (mudança de requisitos, falta de um componente, sobrecarga de tarefas em um ponto do processo, entre outros).

Durante a produção, as fábricas necessitam ter seu fluxo fabril modificado a cada novo requisito de fabricação, demandando alterações significativas na programação, configuração e disposição física de máquinas no ambiente em que operam, o que agrega um alto custo à produção e faz com que as fábricas fiquem paradas por muito tempo.

O uso de sistemas multiagentes vem surgindo como a melhor solução para gerenciar essas modificações no fluxo produtivo, tornando a manufatura auto-organizável, sem a necessidade da interferência humana (LEITÃO, 2013). Agentes como meio de interação entre os elementos da manufatura requerem uma plataforma multiagente, ou *Multi-Agent System* (MAS), definidos em Wooldridge (2009) e descrito em Bellifemine, Caire e Greenwood (2007). MAS é um sistema em que os participantes do processo produtivo podem ser modelados como agentes, onde um agente é uma entidade de software que pode gerenciar cada elemento da manufatura, dispondo suas funcionalidades na forma de serviços. Cada agente pode, então, oferecer ou solicitar serviços dentro de uma plataforma onde os agentes se comunicam uns com os outros. Com estas características os sistemas multiagentes podem fazer com que o sistema produtivo atue com funcionalidades para atender a diversidade, a agilidade e a auto-organização requeridas (BRECHER *et al.*,2020).

Há disponíveis diversas plataformas já desenvolvidas para este propósito, tais como: Jason (BORDINI et al., 2007), Grasshopper (MAGEDANZ *et al.*, 2009), PADE (GREI-UFC, 2019), JADE (WOOLDRIDGE, 2009), entre outras. Cada plataforma está diretamente vinculada à linguagem de programação utilizada em seu desenvolvimento, de forma que a programação dos agentes deve utilizar a mesma linguagem que foi utilizada na plataforma multiagente escolhida.

Este trabalho se propõe a implementar um sistema de manufatura utilizando a plataforma PADE, sendo os agentes programados na linguagem Python e comparar os resultados com a mesma aplicação utilizando

a plataforma JADE, sendo os agentes programados na linguagem Java. Por se tratar de duas linguagens de programação distintas em seus conceitos e códigos, a tomada de decisão na escolha da plataforma segundo métricas definidas é de suma importância, pois uma vez escolhida, o desenvolvimento do sistema assume características de programação específicas. A mudança posterior implica em alterações significativas que causam grandes prejuízos.

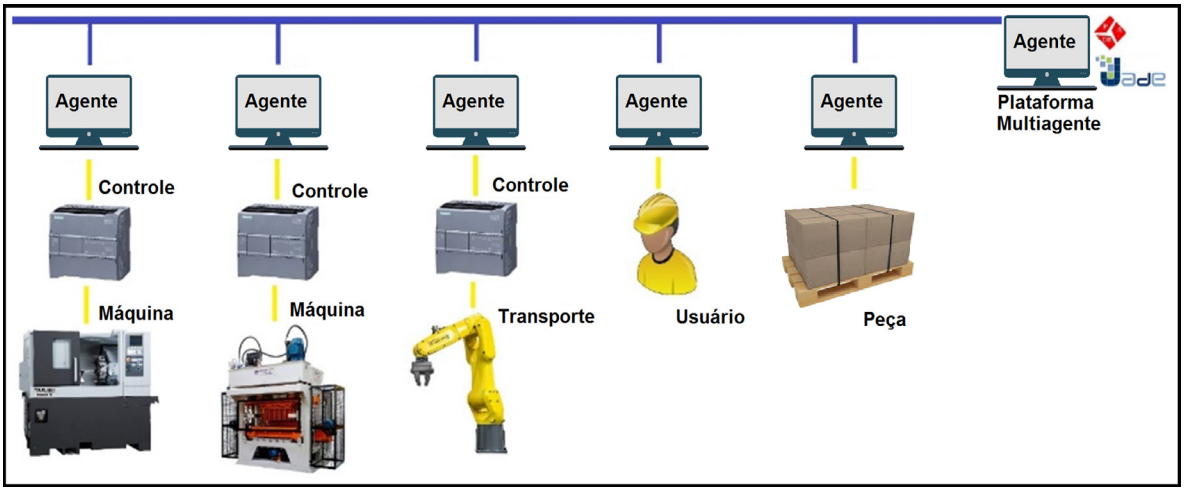
Agentes e sistemas multiagentes

Um agente, segundo Wooldridge (2009), é um sistema de computador autônomo que reside e interage em um determinado ambiente, visando à realização dos objetivos para os quais foi projetado. É afirmado por Sadhu e Konar (2020) que estes agentes devem ter autonomia; porém, devido a se tratarem de entidades de software, devem seguir conceitos e regras, pois são designados a realizar funções específicas em domínios particulares.

Os sistemas multiagentes (MAS – *Multi Agent Systems*), segundo Abbas *et al.* (2015), são coleções de Agentes autônomos situados em um determinado ambiente virtual. Estes agentes comunicam-se entre si, de acordo com as mudanças deste ambiente, com o intuito de conquistar seus objetivos. Também é afirmado que os agentes, analogamente, assumem um papel abstrato, seja uma atividade ou serviço, com diferentes níveis hierárquicos.

É possível comparar agentes operários, com um operador de maquinário em uma fábrica, onde esses têm a capacidade de manipular determinados equipamentos e agentes administradores, como responsáveis pela distribuição dos operários dentro de seu ambiente. Na Figura 1 pode-se observar a configuração de um sistema multiagentes para a indústria.

Figura 1: Configuração de um sistema multiagentes para a indústria.



Fonte: Autores (2022).

O meio de comunicação entre os agentes implementados é regido pela *Foundation for Intelligent Physical Agents* (FIPA), uma organização que promove tecnologias baseadas em agentes e a sua interoperabilidade com outras tecnologias (FIPA, 2002).

O padrão FIPA-*Agent Communication Language* (FIPA-ACL) é um protocolo que abrange cerca de 20 tipos de comunicação, baseados no diálogo entre seres humanos, que agrega algumas propriedades, como: *Performative*, indicador do objetivo da interação do agente; *Content*, o conteúdo da mensagem, análogo ao dito entre uma conversação por seres humanos; *Sender* e *Receiver*, remetente e destinatário da mensagem, respectivamente.

A implementação de agentes de software neste trabalho é realizada a partir do uso de plataformas para o desenvolvimento de sistemas multiagentes. A Plataforma JADE *(Java Agent Development Framework)*, utiliza a linguagem de programação Java no desenvolvimento dos agentes. Nesta plataforma são implementados os protocolos de comunicação definidos pela FIPA na forma de código aberto. Munida de uma interface gráfica de supervisão e controle de agentes, a plataforma conta com vasta documentação e utilização no meio acadêmico

(BELLIFEMINE et al., 2007). As especificações FIPA definem alguns componentes de controle mandatório da plataforma multiagente, entre estes o *Directory Facility* (DF) e o *Agent Management System* (AMS).

O *Directory Facility* é um recurso presente na plataforma JADE que tem a funcionalidade de registrar informações sobre os agentes. O agente posta os seus serviços em tempo de execução e os mesmos ficam disponíveis para consultas por outros agentes.

Já o *Agent Management System* é uma ferramenta que armazena as identificações de cada agente existente no sistema, como o seu endereço e seu nome no universo de agentes. Todo agente deve se conectar ao AMS para receber uma identidade válida, que é utilizada na comunicação entre entidades.

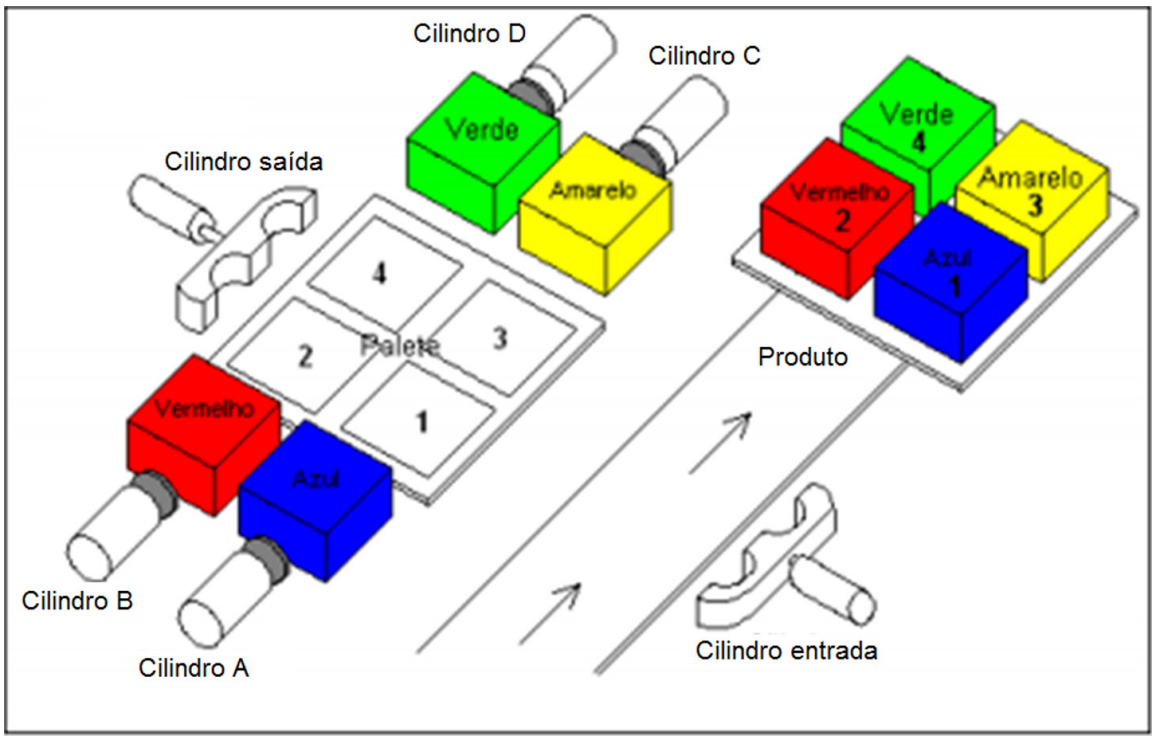
O *Python Agent Development Framework* (PADE) é uma plataforma para o desenvolvimento de sistemas multiagentes, escrito na linguagem de programação Python (KUHNEM, 2018), de código aberto nos termos da licença do *Massachusetts Institute of Technology* - MIT e é desenvolvido no Departamento de Engenharia Elétrica da Universidade Federal do Ceará (GREI-UFC, 2019). Nesta plataforma, todos os agentes que estiverem em execução são capazes de comunicar-se entre si.

Metodologia

O método proposto visa à implementação, através da plataforma multiagente PADE, de um sistema multiagente em um ambiente similar ao fabril, analogamente ao proposto em Peixoto (2012), que efetivou um MAS empregando a plataforma JADE e comparou os resultados. A planta de manufatura que foi implantada com a plataforma JADE no trabalho de Peixoto (2012), ao ser implementada com a plataforma PADE, deve possuir as mesmas funcionalidades e características, de forma que os resultados possam ser comparados, a partir das mesmas métricas.

A proposta conceitual executada foi a de um sistema de manufatura, onde o produto final era um palete que poderia ter combinações de 1 a 4 blocos coloridos em posições determinadas. A montagem dos cubos no palete representa um processo realizado em um ambiente de manufatura industrial, como: pintura, montagem, furação, entre outras. A Figura 2 mostra o croqui do sistema proposto.

Figura 2: Croqui do produto proposto para implementação e comparação dos resultados.



Fonte: adaptado de Peixoto (2012).

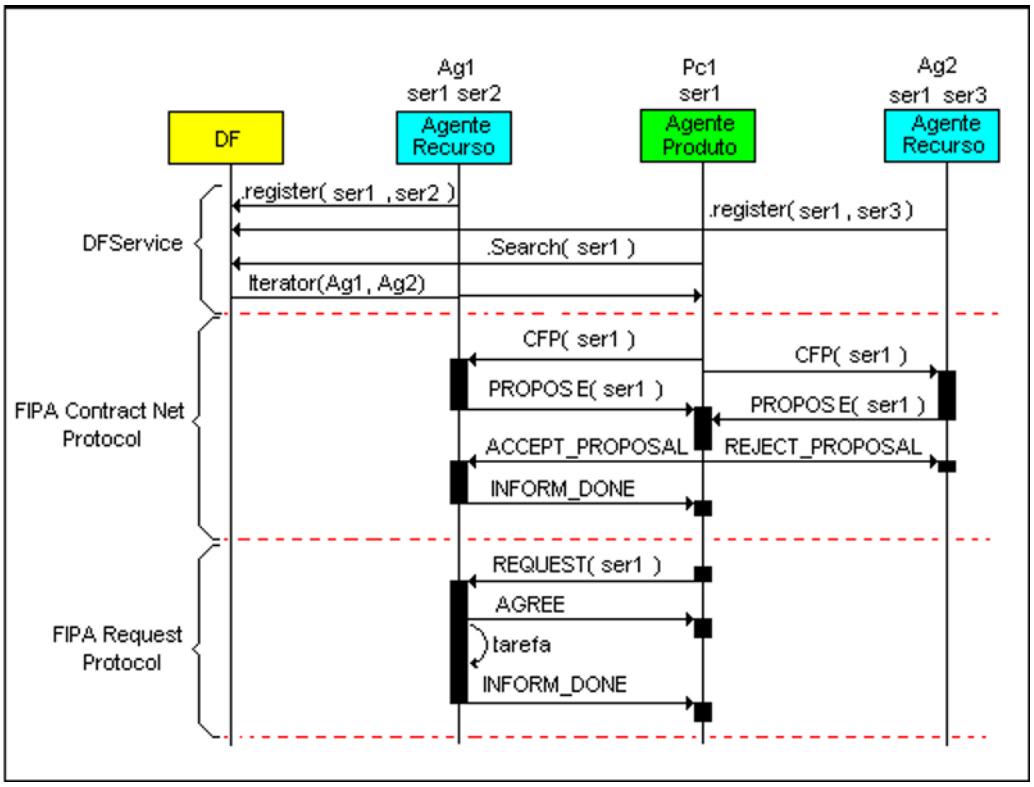
Este trabalho também define que o transporte dos paletes é realizado por uma esteira em um circuito fechado, que passa em frente às estações de montagem. A esteira fica constantemente se locomovendo e a entrada e saída do produto são realizadas pelo operador de forma manual. Logo, não há estação de entrada

nem estação de saída do produto.

Cada estação de trabalho recebe um endereço que corresponde ao local onde a máquina está situada no sistema de manufatura. Com este endereço o produto determina qual o momento que o operador deve lhe inserir na unidade. Este sistema permite adicionar e remover estações de trabalho dinamicamente, e o MAS deve configurar-se, em tempo de execução, para que seja possível continuar operando sem necessidade de parada total do conjunto de manufatura.

A partir das características definidas para esta planta, foi necessária a implementação de agentes de recurso e agentes de produtos. Os agentes de recurso, em primeiro momento após serem instanciados, realizam seu cadastro no DF e tornam-se disponíveis para que outros agentes possam pesquisá-los. Quem faz esta pesquisa são os agentes de produtos, que solicitam os serviços necessários para a confecção da peça. O DF então concede ao solicitante uma lista com todas as estações que realizam este processo e com tal informação, o agente-peça realiza as seguintes etapas de negociação utilizando os protocolos FIPA *ContractNet* e FIPA *Request*, descritos na Figura 3.

Figura 3: Diagrama de sequência de comunicação entre agentes.



Fonte: Autores (2022).

Como métricas, serão analisados conceitos qualitativos e quantitativos em busca de comparar os resultados entre as plataformas PADE e JADE, conforme os apresentados por Peixoto (2012). O Quadro 1 exibe as métricas utilizadas, junto a sua justificativa técnica, forma de obtenção e critério para avaliação.

Quadro 1: Métricas da avaliação.

Métricas	Critério para avaliação
Variação do tempo de produção com aumento estações.	O acréscimo no tempo de produção em função do aumento das estações deve ser o menor possível.
Manter a produção com a retirada de estações.	Saindo uma das estações, outra deve assumir a tarefa, sem requerer alteração na programação.
Manter a produção com a inserção de novas estações.	inserção de uma nova estação não deve requerer alterações na programação e nem demandar parada no sistema.
Quantidade de memória ocupada pelo programa para controlar todo o sistema.	Quanto menor a quantidade de memória ocupada pelo sistema, melhor o seu desempenho.
Competências necessárias para operar e implementar as alterações no sistema de montagem.	Quanto menor o conhecimento requerido, mais fácil de se aplicar.
Linhas de código de programação	O número de linhas de código deve ser o menor possível.
Complexidade do algoritmo para programação	Número de métodos disponibilizado pela plataforma, utilizado na implementação

Para implementação, os agentes de software serão desenvolvidos empregando a plataforma PADE, integrados em computadores, e se comunicarão através da rede local, pelo protocolo TCP/IP. Para a realização dos testes, os computadores deverão ter a mesma versão da plataforma, o mesmo sistema operacional e ter os mesmos componentes de Hardware. Nesta implementação foram utilizados 3 computadores notebook Lenovo, processador Intel Core i3, 64 bits, frequência de trabalho de 2,3GHz, 8Gb de memória temporária, com sistema operacional Windows 8.1.

Implementação do sistema utilizando JADE

No trabalho de Peixoto (2012) os agentes de transporte não são implementados, pois a esteira não requer um gerenciamento de acionamento, ou seja, não haveria serviços que ela pudesse dispor.

Relacionando os agentes que irão compor a plataforma, tem-se o Agente de Recursos, agente de montagem, responsável pela montagem de cubos sobre o produto, e o Agente de Produto, agente-peça, responsável pela busca e alocação dos serviços necessários para executar o fluxo de montagem requerido. Ambos são implementados a partir da classe Agent, disponível na plataforma JADE. Tanto o agente-peça (PA) quanto o agente-montagem (RA) utilizarão uma classe chamada *Utils*, que fornece métodos para postagem dos serviços e pesquisa destes (ABBAS; SHAHEEEN; AMIN, 2015). A primeira ação do agente-peça é buscar no DF a relação de agentes que realiza os serviços desejados, e a partir deste ponto começam as negociações para alocar os recursos necessários. No agente de montagem destaca-se o comportamento "TarefaSimuladaTX", que é o módulo de hardware e que fará o contato com os dispositivos de acionamento e sensoriamento propriamente dito.

Implementação do sistema utilizando PADE

Para a realização dos testes, foram utilizados 3 computadores rodando sistema operacional Windows 8.1, conectados em uma rede local, TCP/IP, com o *switch* modelo FS108, da marca Netgear. A versão do PADE utilizada foi a 2.1.2 com a *branch* criada pela Universidade Federal do Pará, disponível no Github[1].

Na implementação dos agentes empregando o PADE, o framework não conta originalmente com o DF, peça central para o funcionamento do sistema, que permite que os agentes registrem seus serviços para serem posteriormente encontrados na negociação do uso de recursos. Contudo, o PADE fornece a infraestrutura

[1]Endereço do repositório: https://github.com/italocampos/pade

básica que permite o seu desenvolvimento. O desenvolvimento do DF foi realizado com a construção de 3 classes: *Directory*, Registro e *Dfacility*.

A classe Registro contém métodos para registrar e cancelar o registro de um agente, bem como buscar por um determinado serviço, sob controle da classe *Dfacility*. O agente *sniffer*, responsável pela captura de dados de comunicação entre agentes e presente no framework, necessitou de desenvolvimento para apresentar funcionamento adequado no sistema Windows. Foram desenvolvidos dois tipos de agentes: o agente-peça, o agente-estação, com similar correspondência aos agentes utilizados em Peixoto (2012). O agente-estação é capaz de registrar seus serviços no DF e realizar procedimentos sobre os agentes-peça. O agente-peça é capaz de controlar a execução de uma lista de procedimentos para a manufatura do produto, e buscar no DF por agentes capazes de realizar tais operações, utilizando as diretivas do protocolo FIPA *Contract Net* com os agentes máquina.

A comunicação entre o DF e as estações, no momento do registro de serviços, é baseada no protocolo FIPA *Request* (FIPA, 2002). O agente-estação, ao ser instanciado, envia uma mensagem com a performativa INFORM para o agente DF, e como conteúdo os serviços a serem postados. O agente DF armazena estes serviços, juntamente ao nome da estação que foi instanciada e, então, envia uma mensagem com a performativa INFORM_DONE.

Após o registro do agente-estação, os agentes-peça podem solicitar ao DF os serviços que necessitam ser realizados, através da performativa REQUEST e conteúdo, e recebe um INFORM do DF com os identificadores das estações. Nota-se a necessidade de instanciar o agente DF antes de qualquer outro agente de recurso ou produto, uma vez que este é o único que armazena os dados das estações.

Optou-se por utilizar *Python Threads* para continuar a comunicação durante o período ocupado da estação, resultando na eliminação da necessidade de término do contrato via FIPA *Contract Net*.

Resultados e Discussões

A alteração no tempo por ciclo de montagem foi mais elevada na plataforma em Python, possivelmente devido ao framework ainda estar em desenvolvimento. Os resultados são relacionados na Tabela 1.

Tabela 1: Tempo de montagem com acréscimo de estações em JADE e PADE.

Número de estações	Variação com acréscimo de estações (em s) - JADE			Variação com acréscimo de estações (em s) - PADE		
1	26,72	-		52,36	-	
2	27,33	2,28%		53,46	2,10%	
3	28,26	3,40%	16,4% diferença ente 1 e 6 estações	54,43	1,81%	9,49% diferença ente 1 e 6 estações
4	28,82	1,98%		55.18	1,37%	
5	29,87	3,64%		56,56	2,50%	
6	31,10	4,12%		57,33	1,36%	

A métrica manter a produção com a retirada de estações obteve como resultado a comprovação da capacidade do sistema modelado em PADE de continuar operando, relatado na Tabela 2, mesmo com a eliminação de agentes-máquina em tempo de execução, de modo análogo a métrica manter a produção com a inserção de estações, que também se confirmou verdadeira.

Tabela 2: Comparação entre os comportamentos com inserção e retirada de estações.

Estação inserida ou retirada	JADE	PADE
1	O sistema passou a ser montado pelas estações 2 e 3. Não houve necessidade de reprogramação. Na inserção passou a ser mais uma opção de montagem.	O sistema passou a ser montado pelas estações 2 e 3. Não houve necessidade de reprogramação. Na inserção passou a ser mais uma opção de montagem.
2	O sistema passou a ser montado pelas estações 1 e 3. Não houve necessidade de reprogramação. Na inserção passou a ser mais uma opção de montagem.	O sistema passou a ser montado pelas estações 1 e 3. Não houve necessidade de reprogramação. Na inserção passou a ser mais uma opção de montagem.
3	O sistema passou a ser montado pelas estações 1 e 2. Não houve necessidade de reprogramação. Na inserção passou a ser mais uma opção de montagem.	O sistema passou a ser montado pelas estações 1 e 2. Não houve necessidade de reprogramação. Na inserção passou a ser mais uma opção de montagem.

As competências necessárias para operar e implementar o sistema são também iguais, por se tratarem de plataformas similares, o que aponta que os operadores devem ter conhecimentos sobre utilização de sistemas operacionais, além dos já necessários para operar sistema de cunho industrial, conforme apontado na Tabela 3.

Tabela 3: Comparação entre as competências necessárias para operar os sistemas analisados.

Atividades	JADE	PADE
Ligar e preparar o sistema de montagem para o trabalho	operar sistemas industriais; operar sistema operacional de computador;	operar sistemas industriais; operar sistema operacional de computador;
Solicitar montagens de produtos.	operar sistemas industriais; operar sistema operacional de computador;	operar sistemas industriais; operar sistema operacional de computador;
Retirar estações de montagem no sistema.	operar sistemas industriais; operar sistema operacional de computador;	operar sistemas industriais; operar sistema operacional de computador;
Inserir estações de montagem no sistema.	operar sistemas industriais; operar sistema operacional de computador;	operar sistemas industriais; operar sistema operacional de computador;

A quantidade de memória utilizada em JADE com 6 estações é 3.628 vezes menor que com uma estação em PADE. A diferença entre memória para cada implementação também aumenta em uma razão muito maior que a encontrada na primeira plataforma. Mesmo que o número de linhas de códigos tenha sido semelhante, com mesmo comportamento, uma vez que ambas linguagens Python e JAVA geram objetos instanciáveis.

Quanto a métrica de complexidade do algoritmo, a plataforma JADE apresenta mais métodos prontos para serem utilizados no programa reduzindo a complexidade do algoritmo. Na plataforma PADE há necessidade de implementar métodos para garantir funcionalidades, como o caso do DF *(Directory Facilitator)* que em PADE deve ser implementado, tornando o algoritmo mais complexo. Os resultados são encontrados na Tabela 4.

Tabela 4: Comparação entre o uso de memória e linhas de códigos das plataformas.

	JADE			PADE		
Estação Inserida	Memória (em KBs)	Normaliza-da	Linhas de código	Memória (em KBs)	Normaliza-da	Linhas de código
1	11,87	1,000	620	71.744	1,000	471
2	11,25	1,034	620	122.012	1,700	471
3	12,55	1,154	620	171.680	2,393	471
4	14,88	1,368	620	221.348	3,085	471
5	15,96	1,467	620	271.016	3,777	471
6	19,77	1,818	620	320.684	4,469	471

Como pontos a discutir, se pode apontar as métricas de inserção e remoção de estações em tempo de execução, cujos resultados demonstram-se de elevada aplicabilidade para a atualização dos sistemas de manufatura não auto-organizáveis.

O resultado com maior atenuação negativa à implicação é a utilização de memória da plataforma PADE, em que o uso é consideravelmente maior que o em JADE; todavia, os requisitos de *hardware* para implementar a plataforma são considerados baixos perto da capacidade computacional popularmente encontrada, no caso onde os agentes de *software* são inseridos em diferentes computadores. Acredita-se que a utilização de um ambiente virtual de desenvolvimento, indicado pelo grupo desenvolvedor da plataforma, acarretou em uma utilização maior de memória.

Considerações Finais

Frente às novas exigências de um mercado de consumidores cada vez mais exigentes, as empresas precisam buscar meios para continuar existindo numa indústria que é cada vez mais competitiva. A automação industrial é a principal ferramenta utilizada hoje nas empresas mais bem sucedidas e os sistemas multiagentes se mostram como mais uma robusta adição a essa ferramenta.

Este trabalho apresentou a modelagem de um processo de manufatura empregando sistemas multiagentes em busca de um sistema adaptável às mudanças no ambiente, causadas por alterações no processo produtivo originadas na demanda diversificada de produtos ou mesmo falhas em equipamentos. O uso de agentes neste processo mostrou-se vantajoso para o desenvolvimento, uma vez que tanto a computação quanto a comunicação ficam padronizadas pelo emprego de tipos de comportamentos e pelo uso de padrões de comunicação bastante específicos que estão presentes no paradigma orientado a agentes.

O desenvolvimento de sistemas multiagentes empregando frameworks dedicados a esta atividade reduzem o tempo necessário para a sua criação, uma vez que fornecem uma infraestrutura básica de comunicação e primitivas para a troca de mensagens. Da mesma forma, a padronização dos protocolos de comunicação permite que os procedimentos desenvolvidos para a negociação e execução de tarefas entre produtos e estações durante o processo de manufatura, visando à otimização da produção, sejam reaproveitados no desenvolvimento do sistema em diferentes linguagens de programação. O desenvolvedor também tem seu trabalho facilitado por não necessitar entrar em detalhes específicos de interações entre diferentes dispositivos via rede. Neste caso, o esforço de aprendizado fica reduzido às particularidades dos frameworks utilizados, reaproveitando tanto o conhecimento sobre a construção de agentes quanto o conhecimento sobre suas formas de comunicação.

A modelagem do sistema de manufatura empregando multiagentes foi realizada com o objetivo de testar a robustez e desempenho do sistema resultante face às mudanças propostas no ambiente, tendo métricas específicas para a sua avaliação. O uso de dois frameworks e duas linguagens distintas demandou a escolha de métricas de comparação que permitissem avaliar as ferramentas escolhidas. Nos testes realizados, Python mostrou-se uma linguagem de fácil aprendizado e considerando-se o desenvolvimento de sistemas multiagentes, com um código mais compacto quando comparada a Java. Como desvantagem, o uso de memória por parte da implementação em Python mostrou-se muito maior do que por parte da implementação em Java.

Foram, assim como o esperado, alcançados os objetivos qualitativos em relação às métricas. O PADE, mesmo se tratando de uma plataforma mais nova e menos utilizada que a, já muito documentada, JADE, pode se tornar uma alternativa para implementação devido ao cenário de ascensão da linguagem Python pela comunidade de desenvolvedores.

O uso de sistemas multiagentes, seguindo a abordagem apresentada em Peixoto (2012), mostrou um potencial para se desenvolver um sistema bastante robusto, tornando possível a modelagem de sistemas de manufatura flexíveis que mostram adaptação mesmo em face às variações no custo de produção originadas dinamicamente, bem como alteração na quantidade de recursos produtivos. Nos experimentos foram adicionados e subtraídos recursos, e comprovou-se as alterações correspondentes no processo de produção, mantendo a sua execução com baixo impacto sobre o desempenho. Como trabalhos futuros fica a melhoria da interface gráfica do sistema e a inclusão de agentes de transporte no processo de produção.

Referências

ABBAS, H.A.; SHAHEEN, S. I.; AMIN. M. H. Organization of Multi-Agent Systems: An overview. **International Journal of Intelligent Information Systems**, v.4, n.3, p, 46-57, 2015. Disponível em: https://arxiv.org/ftp/arxiv/papers/1506/1506.09032.pdf. Acesso em: 03 dez. 2019.

BELLIFEMINE, F. L.; CAIRE, G.; GREENWOOD, D. **Developing Multi-Agent Systems with JADE**. Nova Iórque: John Wiley & Sons, 2007.

BRECHER, C. et al. Simulation framework for virtual robot programming in reconfigurable production systems. **Procedia CIRP,** v. 86, p. 98–103, 2020. Disponível em: https://doi.org/10.1016/j.procir.2020.01.045. Acesso em 12 dez. 2020.

BORDINI, Rafael H.; HÜBNER, Jomi Fred; WOOLDRIDGE, Michael. **Programming Multi-Agents Systems in AgentSpeak using Jason**. Chichester: John Wiley, 2007.

FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS - FIPA. **FIPA Communicative Act Library Specification**. Geneva, Switzerland. 2002.

GREI-UFC. **Python Agent Development framework, 2019**. Disponível em: https://pade-docs-en.readthedocs.io/en/latest/. Acesso em: 30 nov. 2019.

LEITÃO, Paulo. Multi-agent systems in industry: current trends & future challenges. **Conference Beyond AI: Interdisciplinary Aspects of Artificial Intelligence**. Pilsen: Springer. p. 197-201, 2013. Disponível em: http://hdl.handle.net/10198/9677. Acesso em: 30 mai. 2020.

MAGEDANZ, Thomas et al. **Grasshopper:** A Universal Agent Platform Based on OMG MASIF and FIPA Standards. 2009. Disponível em: http://cordis.europa.eu/infowin/acts/analysys/products/thematic/agents/ch4/ch4.htm. Acesso em: 26 jun. 2020.

PÉREZ, Marta Pérez; BEDIA, Ana María Serrano; FERNÁNDEZ, María Concepción López. A review of manufacturing flexibility: systematising the concept. **International Journal Of Production Research,** [S.L.], v. 54, n. 10, p. 3133-3148, 28 jan. 2016. Disponível em: https://doi.org/10.1080/00207543.2016.1138151. Acesso em: 12 jun. 2020.

KOTLER, P.; KARTAJAYA, H.; SETIAWAN, I. **Marketing 4.0: do tradicional ao digital**. Rio de Janeiro: Sextante, 2017.

KUHNEM, Matheus. **Python: a linguagem que está engolindo o mercado**, 2018. Artigo em site. Disponível em: https://medium.com/tendências-digitais/python-engolindo-o-mercado-6872769800b2. Acesso em: 2 dez. 2019.

PEIXOTO, J. A. **Desenvolvimento de sistemas de automação da manufatura usando arquiteturas orientadas a serviço e sistemas multiagentes.** Dissertação (Mestrado) – Departamento de engenharia elétrica, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2012. Disponível em: https://www.lume.ufrgs.br/bitstream/handle/10183/61385/000865282.pdf?sequence=1. Acesso em: 03 dez. 2019.

SADHU, Arup Kumar; KONAR, Amit. **Multi-Agent Coordination:** a reinforcement learning approach. Piscataway: Wiley-Ieee Press, 2020. 310 p.

WOOLDRIDGE, Michael. **An Introduction to MultiAgent Systems.** 2 ed. Chichester: John Wiley & Sons, 2009.