Knowledge Graphs for Decoupling Intelligent Agents in Web of Things-Based Industrial Systems

Juliano Q. Nunes[1], Iderli P. Souza Filho[1], Fernando R. Santos[1], Jomi F. Hübner[1]

[1]Departamento de Automação e Sistemas (DAS)
Universidade Federal de Santa Catarina (UFSC) – Florianópolis – SC – Brazil

julianoqnunes17@gmail.com, fernando.rod.santos@gmail.com

iderli.souza@posgrad.ufsc.br, jomi.hubner@ufsc.br

Abstract. *The increasing demand for flexibility and interoperability in industrial systems highlights the limitations of approaches in which intelligent agents rely on specific sensor and actuator identifiers to perform their tasks. Although the Web of Things (WoT) standard provides mechanisms for describing and accessing devices in a standardized manner, agents remain coupled to the physical infrastructure when their decisions depend on prior knowledge of resource-specific identifiers. This paper presents a Knowledge Graph-based architecture to decouple intelligent agents from industrial devices in WoT environments. The approach was validated using the station of a FESTO manufacturing line, responsible for inspecting the orientation of workpieces. The station was modeled in RDF using standardized ontologies to represent both physical components and their functional capabilities. As a result, agents can discover resources based on their functions rather than relying on device-specific identifiers. The results demonstrate that infrastructure modifications can be accommodated through updates to the Knowledge Graph without requiring changes to the agent's logic. The main contribution of this work is the replacement of device-based discovery with capability-based discovery, promoting greater flexibility, interoperability, and reusability in industrial automation systems.*

Resumo. *A crescente necessidade de flexibilidade e interoperabilidade em sistemas industriais evidencia as limitações de abordagens nas quais agentes inteligentes dependem de identificadores específicos de sensores e atuadores para executar suas tarefas. Embora o padrão Web of Things (WoT) forneça mecanismos padronizados para descrever e acessar dispositivos, os agentes ainda permanecem acoplados à infraestrutura física quando suas decisões dependem do conhecimento prévio dos recursos disponíveis. Este trabalho apresenta uma arquitetura baseada em grafos de conhecimento para desacoplar agentes inteligentes dos dispositivos industriais em ambientes WoT. A abordagem foi validada por meio da estação de uma linha de manufatura da FESTO, responsável pela inspeção da orientação de peças. A estação foi modelada como um grafo de conhecimento em RDF utilizando ontologias padronizadas para representar tanto os componentes físicos quanto suas capacidades funcionais. Dessa forma, o agente passa a descobrir recursos com base em suas capacidades funcionais, em vez de depender de identificadores específicos. Os resultados demonstram que modificações na infraestrutura podem ser acomodadas por meio da atualização do grafo de conhecimento, sem alterações na lógica do agente.*

1. Introdução

À medida que os sistemas industriais evoluem para arquiteturas cada vez mais distribuídas e dinâmicas, torna-se necessário o desenvolvimento de mecanismos que permitam que os agentes operem de forma autônoma em ambientes sujeitos a constantes mudanças. Nesse cenário, os Sistemas Multiagentes (SMA) desempenham um papel importante ao possibilitar a tomada de decisão distribuída e a coordenação entre componentes heterogêneos. Porém, para que esses agentes atuem efetivamente em ambientes web e industriais, é necessário reduzir sua dependência de detalhes específicos da infraestrutura, permitindo que se adaptem a alterações nos recursos disponíveis sem comprometer a execução de seus objetivos.

Segundo [Berners-Lee et al. 2023], a Web Semântica estende a Web tradicional ao adicionar significado explícito aos dados, permitindo que sistemas computacionais interpretem informações e relações de maneira mais próxima à compreensão humana. Essa capacidade possibilita que agentes autônomos utilizem o conhecimento disponível para descobrir recursos, interpretar seu contexto e executar tarefas de forma mais inteligente. Nesse âmbito, a Web Semântica fornece os mecanismos necessários para representar formalmente entidades, propriedades e relacionamentos, constituindo a base para a construção de Grafos de Conhecimento e para a interoperabilidade entre sistemas heterogêneos.

Nesse contexto, a Indústria 4.0 promove a integração entre sistemas físicos e digitais, possibilitando que equipamentos, sensores, softwares e serviços atuem de forma coordenada ao longo dos processos produtivos [Kagermann et al. 2013]. A interoperabilidade torna-se um requisito essencial, uma vez que os diferentes componentes do ambiente industrial precisam compartilhar informações e colaborar de maneira eficiente, mesmo quando desenvolvidos por fabricantes ou tecnologias distintas [Hermann et al. 2016]. Contudo, alcançar essa interoperabilidade continua sendo um desafio, especialmente em cenários dinâmicos nos quais dispositivos podem ser adicionados, removidos ou substituídos com frequência.

Logo, as abordagens baseadas em WoT padronizam a interface de acesso, mas não eliminam necessariamente a dependência do agente em relação à identidade e configuração específica dos recursos. Dessa forma, torna-se necessário desenvolver mecanismos que permitam que aplicações e agentes interajam com os recursos disponíveis de maneira flexível e independente de suas implementações específicas. Assim, investigamos uma arquitetura baseada em tecnologias da Web Semântica e Grafos de Conhecimento para promover a interoperabilidade em ambientes industriais. A proposta busca permitir que agentes descubram e utilizem recursos a partir de suas capacidades funcionais e do significado associado aos dados, em vez de depender de identificadores ou configurações específicas de dispositivos. Com isso, pretende-se reduzir o acoplamento entre a lógica de controle e a infraestrutura física.

2. Referencial Teórico

2.1. Grafos de Conhecimento

Os Grafos de Conhecimento representam informações complexas por meio de entidades (nós) conectadas por relações semânticas (arestas), formando redes navegáveis de dados interconectados. Conforme [Ji et al. 2021], estas estruturas caracterizam-se por sua

natureza heterogênea, extensibilidade incremental e capacidade de capturar incerteza e proveniência por meio de anotações.

As vantagens dos grafos na representação de relações complexas incluem a modelagem natural de relações n-árias e hierárquicas, facilidade de integração de fontes heterogêneas e possibilidade de inferência por meio de raciocínio automatizado. A natureza visual dos grafos também facilita a compreensão humana e mantém o processamento computacional eficiente [Ji et al. 2021].

Em contextos industriais, os grafos de conhecimento são aplicados na manutenção preditiva, integrando dados de sensores, históricos de manutenção e especificações técnicas para identificar padrões e prever falhas. Na otimização de cadeias de suprimentos, representam relações complexas entre fornecedores, produtos e processos, e na gestão de qualidade, conectam parâmetros de processo com resultados de inspeção. [Buchgeher et al. 2021] demonstram como essas aplicações reduzem o tempo de inatividade e melhoram a eficiência operacional em ambientes de manufatura.

2.2. RDF e a Linguagem Turtle

O Resource Description Framework (RDF) constitui o modelo de dados fundamental da Web Semântica, fornecendo uma estrutura flexível e extensível para representar informações sobre recursos de forma processável por máquinas. Desenvolvido pelo World Wide Web Consortium (W3C), o RDF baseia-se em um modelo de grafos direcionados que permite expressar relações entre recursos por meio de uma sintaxe uniforme [Cyganiak et al. 2014].

O modelo de dados RDF fundamenta-se no conceito de triplas, nas quais cada declaração consiste em três componentes: *sujeito* (recurso sobre o qual se faz uma afirmação), *predicado* (propriedade ou relação) e *objeto* (valor ou outro recurso relacionado). Esta estrutura simples permite a construção incremental de grafos de conhecimento complexos [Klyne 2004].

O Turtle (*Terse RDF Triple Language*) é uma sintaxe compacta e legível para serialização de grafos RDF, utilizando prefixos para abreviar URIs e atalhos sintáticos para padrões comuns [Beckett and Berners-Lee 2011]. O Código 1 apresenta a descrição de um sensor de temperatura em Turtle.

Código 1. Exemplo de descrição RDF em Turtle

```
@prefix td: <https://www.w3.org/2019/wot/td#>.
@prefix sosa: <http://www.w3.org/ns/sosa/>.

:sensor01 a td:Thing, sosa:Sensor;
    td:title "Sensor de Temperatura Industrial"@pt;
    sosa:observes :temperatura_sala_101;
    td:hasPropertyAffordance [
        td:name "temperatura";
        td:hasForm [
            td:href "mqtt://broker.local/temp01"
        ]
    ].
```

3. Trabalhos Relacionados

Os paradigmas de controle baseados em agentes para automação industrial compartilham características de autonomia local, comunicação *peer-to-peer* e emergência de comportamento global a partir de interações locais [Leitão 2009]. Esses sistemas distribuem inteligência e tomada de decisão, permitindo que cada recurso de produção tome decisões locais, aumentando robustez e escalabilidade. Essa descentralização facilita o paradigma *plug-and-produce* para integração de novos equipamentos [Monostori et al. 2016].

A integração com dispositivos físicos ocorre por meio de agentes *wrapper* que encapsulam funcionalidades de CLPs (Controladores Lógicos Programáveis), sensores e atuadores. Protocolos como OPC-UA e WoT fornecem abstrações para a integração de dispositivos heterogêneos, embora persistam desafios de interoperabilidade semântica entre diferentes modelos de informação [Leitão et al. 2016]. Nesse contexto, as abordagens que combinam agentes com ontologias e tecnologias da Web Semântica têm sido investigadas como forma de permitir um entendimento compartilhado sobre capacidades, estados e comportamentos de dispositivos [Monostori et al. 2016].

O trabalho de [Szilagyi and Wira 2016] apresenta uma visão geral sobre o uso de ontologias no contexto da Internet of Things (IoT), destacando o papel da padronização na descrição de dispositivos, serviços e dados. Os autores argumentam que as representações semânticas aumentam a interoperabilidade entre sistemas heterogêneos e facilitam o processamento automatizado de informações por aplicações inteligentes.

De forma complementar, [Zou et al. 2003] investigam a integração entre tecnologias da Web Semântica e Sistemas Multiagentes. Os autores demonstram como linguagens como RDF e OWL podem ser utilizadas em conjunto com padrões da FIPA para permitir que agentes compartilhem conhecimento de maneira estruturada e interoperável, fortalecendo a cooperação em ambientes dinâmicos baseados em informação distribuída.

A Hypermedea representa uma extensão da plataforma JaCaMo voltada ao desenvolvimento de agentes capazes de atuar em ambientes Web e WoT. A integração com a Web Semântica ocorre por meio de artefatos CArtAgO especializados, de modo a permitir a navegação por recursos RDF, o raciocínio sobre o conhecimento semântico e a interação com Thing Descriptions (TD) [Kaebisch et al. 2023], por meio de propriedades, ações e eventos dos dispositivos WoT. A combinação entre o raciocínio simbólico do agente e o conhecimento semântico posiciona a Hypermedea como solução para agentes em ambientes ciberfísicos semanticamente anotados [Charpenay et al. 2022].

Portanto, apesar dessas abordagens oferecerem mecanismos para integração, interoperabilidade e representação semântica de recursos, permanece o desafio de permitir que agentes descubram recursos com base em suas capacidades funcionais, sem depender de identificadores específicos da infraestrutura.

4. Desenvolvimento do Projeto

Este trabalho foi desenvolvido no Laboratório de Automação e Informática Industrial (LAI), da Universidade Federal de Santa Catarina (UFSC). O código-fonte e os artefatos desenvolvidos neste trabalho estão disponíveis publicamente em um repositório GitHub[1].

[1] `https://github.com/juliano-quatrin-nunes/agents-in-charge/tree/pfc`

O projeto utilizou uma bancada didática de manufatura da FESTO, controlada por um CLP da Siemens, o qual integra diversos sensores e atuadores. Esta bancada contém uma estrutura baseada em padrões da WoT e tem os seus recursos disponíveis em Thing Descriptions (TD), as quais podem ser acessadas via HTTP.

O objetivo deste projeto foi desenvolver e validar uma arquitetura que utilize a camada de conhecimento para o desacoplamento entre a lógica do agente e os dispositivos físicos. O desenvolvimento também definiu uma estrutura de criação automatizada da camada de conhecimento, estabelecendo a integração entre Thing Description e Grafos de Conhecimento.

4.1. Descrição da Linha de Manufatura e da Arquitetura

O estudo de caso utiliza a estação *Separating* de uma linha de manufatura da FESTO, conforme a Figura 1. Ela foi modelada em três subsistemas lógicos, baseados em suas responsabilidades funcionais:

- **Esteira principal** (*MainConveyor*): responsável pelo transporte da peça, abrange a esteira principal (8), o sensor de entrada (1) e o sensor de saída (7);
- **Sistema de verificação de orientação** (*OrientationVerificationSystem*): encapsula a lógica de inspeção, abrange o sensor de orientação (3), o sensor de peça parada (2) e a trava para inspeção (4);
- **Sistema de descarte** (*DiscardSystem*): gerencia o processo de descarte, abrange o atuador de descarte (6), a esteira de descarte (9) e o sensor de descarte (5).

Essa estação tem como função verificar a orientação das peças que passam pelo processo produtivo, identificando aquelas que se encontram invertidas e realizando seu descarte antes que avancem para as etapas seguintes. Para executar essa tarefa, a estação dispõe de sensores responsáveis pela detecção das características da peça e de atuadores que permitem direcionar ou descartar itens de acordo com o resultado da inspeção. O controle desses dispositivos é realizado por um CLP Siemens S7-1200, que disponibiliza suas funcionalidades para as camadas superiores do sistema.

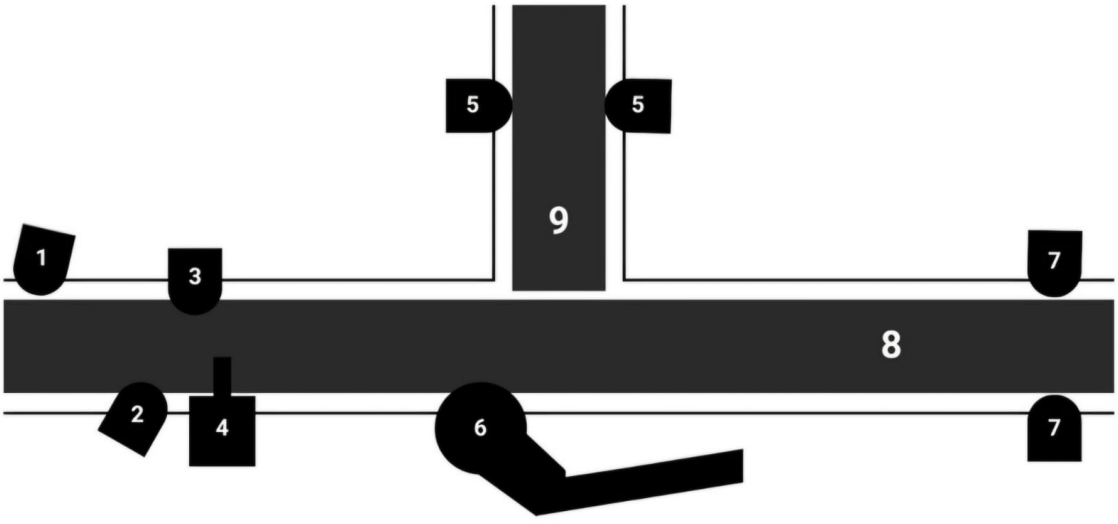

Figura 1. Esquemático da bancada Separating com componentes enumerados.

A solução parte de uma estrutura prévia do ambiente modelado conforme os padrões da WoT. A estrutura inicialmente continha um documento JSON, seguindo os princípios da TD, para a bancada *Separating*, formalizando suas *affordances*. A estrutura da TD seguiu padrões da WoT, utilizando *properties* para representar os sensores e o estado dos atuadores, e *actions* para modelar as operações que poderiam ser executadas.

A arquitetura proposta para introduzir a camada de conhecimento semântico ao projeto e alcançar este desacoplamento foi estruturada em quatro camadas distintas e interconectadas, que separam as responsabilidades desde a interação física até a tomada de decisão autônoma, conforme ilustrado na Figura 2. Na base, a **Camada Física** compreende o hardware da linha de manufatura FESTO, e a **Camada de WoT** atua como sua interface digital, expondo as funcionalidades dos dispositivos por meio de uma API REST, mantendo a infraestrutura de comunicação já validada no projeto anterior.

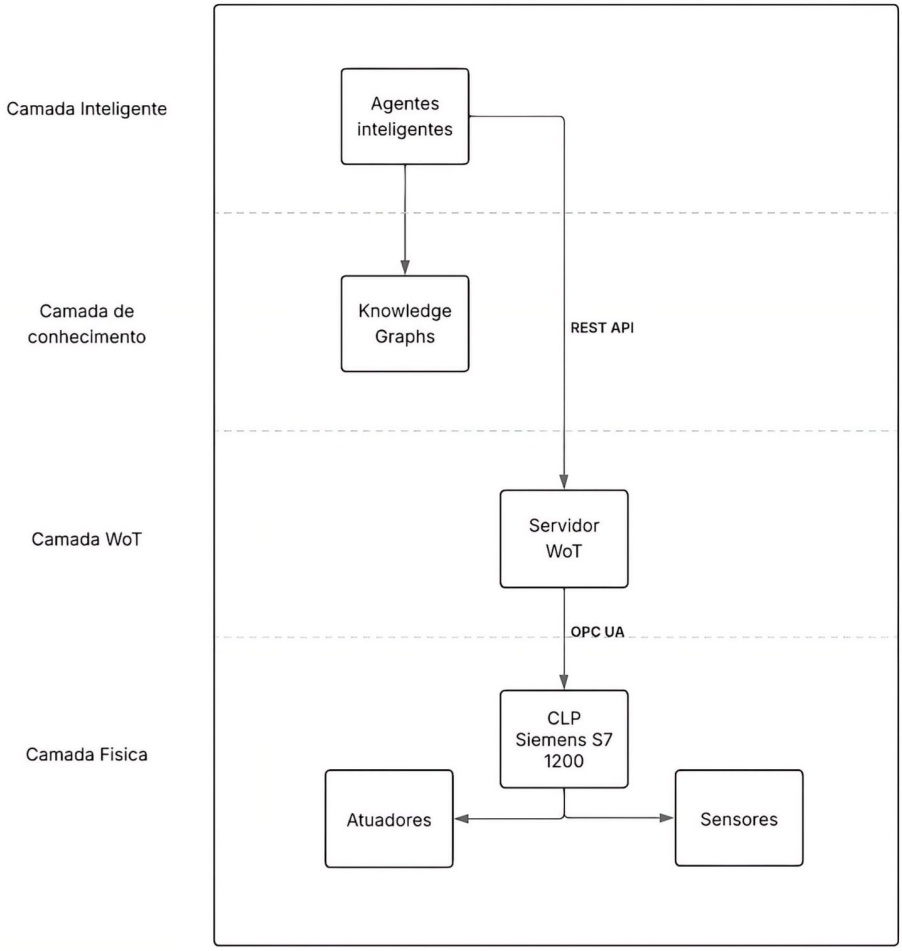

Figura 2. Arquitetura da Solução Proposta

O componente inovador da solução proposta reside nas duas camadas superiores. A **Camada de Conhecimento** é responsável por manter e expor o grafo de conhecimento em RDF. Este grafo descreve semanticamente a planta, suas capacidades funcionais e as informações de interação, servindo como uma fonte única de informação sobre o ambiente. No topo, a **Camada Inteligente** contém o agente BDI, no qual a lógica sequencial de controle está programada, tomando como base as informações de ambiente obtidas.

Nesta arquitetura, o agente na Camada Inteligente inicia um ciclo de descoberta ao consultar a Camada de Conhecimento. Primeiramente, o agente é programado com um plano de controle a ser seguido. A partir do plano, ele busca por dispositivos que possuam a capacidade funcional necessária e extrai os detalhes de sua interface de interação. De posse dessas informações, o agente envia uma requisição HTTP à Camada WoT, que, por sua vez, as traduz em comandos para a Camada Física. O resultado da operação é retornado pela mesma cadeia de comunicação, permitindo que o agente atualize suas crenças e continue seu ciclo deliberativo, completando assim o controle do processo.

4.2. Arquitetura da Camada de Conhecimento

A Camada de Conhecimento foi implementada utilizando o framework Apache Jena e projetada de forma modular, separando as responsabilidades de construção e de exposição do grafo. A Figura 3 ilustra os componentes lógicos principais desta arquitetura: **Construtor Semântico** (*Semantic Builder*) e **Servidor de Linked Data** (*Linked Data Server*).

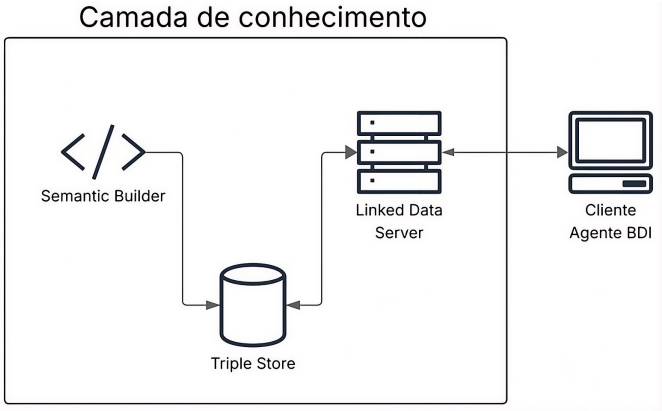

Figura 3. Arquitetura da Camada de Conhecimento

O Construtor Semântico é o componente responsável por criar o grafo de conhecimento. Sua função é instanciar as representações dos componentes da planta e representá-los em triplas RDF de acordo com as ontologias definidas na Interface Semântica, que padroniza a comunicação entre o Grafo de Conhecimento e o agente. Essa interface integra quatro ontologias: *SOSA/SSN*[2] para modelagem da estrutura física e hierárquica dos dispositivos; *SAREF*[3] para descrição da funcionalidade, comandos e estados; uma ontologia de domínio, a *AI4Industry (AI4I)*, definida e utilizada em [Nardin 2024], para especificidades do setor; e *TD/WoT* para detalhes de interação e protocolo. A decisão de isolar essa lógica em um componente separado permite manter a construção do grafo como um processo controlado e consistente, além de facilitar a sua manutenção por meio de uma API programática em Java.

O Servidor de Linked Data é o componente responsável por expor o grafo armazenado no TDB2 (armazenamento nativo de triplas do Apache Jena) para o mundo externo. Sua função é receber requisições HTTP, consultar os dados solicitados no armazenamento RDF e disponibilizá-los no formato apropriado, de acordo com a negociação

[2] `http://www.w3.org/ns/sosa/`
[3] `https://saref.etsi.org/`

de conteúdo. Esta separação de responsabilidades garante que a lógica de exposição seja desacoplada da lógica de construção, permitindo, por exemplo, que o servidor seja atualizado ou substituído no futuro sem impactar a forma como o grafo é gerado. Os detalhes da implementação destes componentes são apresentados a seguir.

5. Estrutura do Grafo Gerado

Para este projeto, foi adotada uma estratégia de nomenclatura na qual a estrutura da URI reflete a hierarquia física e lógica da planta industrial. Esta abordagem aumenta a legibilidade do modelo e fornece um mecanismo de navegação intuitivo tanto para desenvolvedores quanto para o agente consumidor.

Cada segmento do caminho representa um nível na hierarquia de composição do sistema. Por exemplo, o sensor de orientação receberá uma URI `BASE_URI/MPSFesto/Separating/OrientationSystem/OrientationSensor/`, demonstrando que esse sensor faz parte do subsistema de verificação de orientação e da bancada *Separating*, seguindo a mesma hierarquia apresentada na Figura 4.

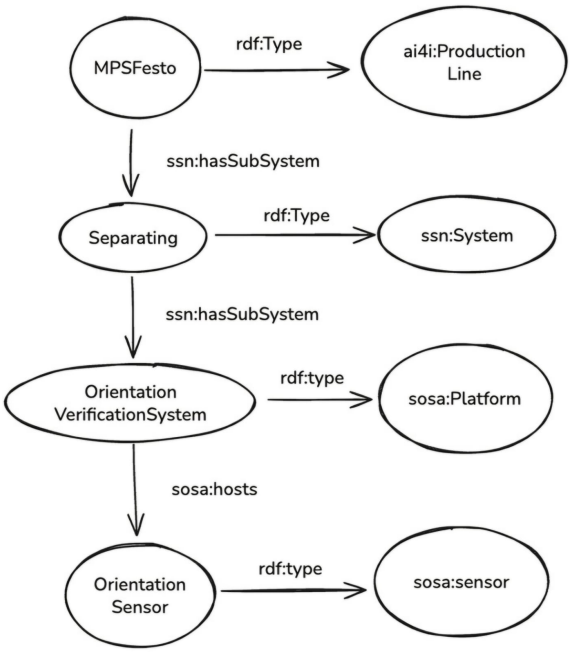

Figura 4. Grafo de conhecimento resumido do sensor de orientação.

Esta decisão de projeto oferece três benefícios principais. Primeiro, a legibilidade humana, que facilita a inspeção, depuração e manutenção do grafo, pois a URI de um recurso revela imediatamente seu contexto e sua posição na estrutura da planta. Segundo, ela permite uma navegabilidade lógica. Um agente que está interagindo com um componente pode, por meio da manipulação da URI (por exemplo, removendo o último segmento), descobrir a URI de seu sistema hospedeiro, permitindo-lhe navegar para níveis superiores da hierarquia para obter informações contextuais. E terceiro, esta estrutura reduz a possibilidade de ambiguidades mediante a definição de escopo, evitando colisões de nomes entre componentes de diferentes bancadas e estabelecendo um *namespace* claro e organizado para toda a planta.

Além disso, esta abordagem de modelagem e nomenclatura reflete diretamente o princípio de design da modularidade. Cada subsistema, como o de verificação de orientação, é modelado como uma entidade autocontida, com suas próprias funções e dispositivos associados. Esta organização modular aumenta a legibilidade e a clareza do modelo, além de facilitar a manutenção e futura extensão do sistema, permitindo que novas funcionalidades ou componentes sejam adicionados de forma controlada e consistente.

6. Desenvolvimento do Agente

O agente de controle foi implementado utilizando a plataforma JaCaMo, que integra o modelo de raciocínio BDI (por meio da linguagem Jason) com um ambiente para artefatos (CArtAgO). A interação com o grafo de conhecimento foi viabilizada pelo framework Hypermedea, que se integra ao CArtAgO e permite que o agente perceba e navegue em ambientes Linked Data.

A lógica do agente foi estruturada como um conjunto de planos hierárquicos, projetados para serem orientados a objetivos de alto nível, como `!processWorkpiece`. A descoberta dos componentes é realizada de forma semântica: o agente busca por um componente com base em suas classes ontológicas e nas relações que ele possui, não em seu identificador. A base de conhecimento do agente é composta por um conjunto de regras lógicas que mapeiam os predicados RDF (por exemplo, `ssn:hasSubSystem` e `saref:accomplishes`) para crenças internas, permitindo que ele raciocine sobre a estrutura e as funções descritas no grafo.

Para cumprir um objetivo, o agente executa uma sequência de consultas lógicas sobre suas crenças para encontrar o recurso apropriado. Um exemplo representativo é a descoberta do atuador de descarte. O plano `!findDiscardDiverter`, que recebe como parâmetro a área de entrada do sistema de descarte (já descoberta semanticamente), segue a seguinte cadeia de inferência para identificar o atuador correto:

1. Descobre qual propriedade está associada à área de entrada;
2. Encontra o comando que atua sobre essa propriedade;
3. Identifica a função que possui esse comando;
4. Finalmente, encontra o atuador que realiza essa função.

Apenas após essa descoberta, o agente obtém a URI do atuador e, em seguida, consulta sua *affordance* de interação. O Código 2 demonstra como esse plano de descoberta foi implementado no agente.

Código 2. Plano para descoberta do desviador de descarte

```
+!findDiscardDiverter(InputArea, DiscardDiverter) <-
    ?hasProperty(InputArea, DiverterStatus)
    & actsUpon(Command, DiverterStatus)
    & hasCommand(DiscardFunction, Command)
    & accomplishes(DiscardDiverter, DiscardFunction).
```

Uma vez que um dispositivo é descoberto, o agente utiliza planos genéricos, como `!readProperty` e `!invokeAction`, que recebem a URI do dispositivo como parâmetro. Estes planos são responsáveis por extrair a *affordance* correspondente do grafo, obter a URL do endpoint e realizar a requisição HTTP. Esta arquitetura de agente, que

separa a lógica de descoberta da lógica de interação, valida o desacoplamento proposto e demonstra a viabilidade de um controle autônomo e robusto a mudanças na infraestrutura física.

7. Validação do Desacoplamento Semântico

Para avaliar experimentalmente a capacidade da arquitetura semântica de reduzir o acoplamento entre o agente e o hardware, foi executado um cenário de teste de adaptabilidade. Este cenário foi projetado para simular uma alteração comum em um ambiente industrial: a substituição de um componente de hardware, que poderia resultar na alteração de sua interface de comunicação, além de sua URI no grafo de conhecimento.

O procedimento de validação foi dividido em duas etapas. Na primeira etapa, o sistema completo foi executado com o grafo de conhecimento original e o comportamento do agente foi registrado para servir como referência. O agente deveria coordenar as etapas do processo de separação de uma peça na esteira. Na segunda etapa, o grafo de conhecimento foi modificado para simular a substituição do desviador de descarte por um atuador que modifica a direção da esteira principal. Com essa alteração, a URI de seu endpoint no middleware WoT foi alterada, além de sua descrição semântica, garantindo que houvesse alinhamento entre o grafo de conhecimento e o ambiente físico. O agente foi então executado novamente, sem qualquer alteração em seu código-fonte. O critério de sucesso foi a capacidade do agente operar a bancada corretamente em ambas as etapas, demonstrando sua adaptação à mudança.

Ao executar novamente o mesmo cenário de teste, o sistema apresentou um comportamento funcional equivalente ao da execução original. O agente foi capaz de descobrir a nova URI ao consultar o grafo pela capacidade funcional do atuador (ou seja, o atuador que `saref:accomplishes` a função `vocab:DiscardFunction`), extrair a nova informação de interação de sua *affordance* e realizar a chamada HTTP para o novo endpoint com sucesso. Esta avaliação experimental evidencia o desacoplamento proporcionado pela arquitetura, demonstrando que a adaptação a mudanças na infraestrutura física atende ao objetivo de reduzir o acoplamento entre agente e hardware.

A análise comparativa evidencia que a arquitetura proposta reduz o acoplamento nominal (dependência de identificadores específicos dos dispositivos) no cenário avaliado, ao substituir a dependência de identificadores estáticos pela descoberta de recursos baseada em capacidades funcionais. A principal vantagem desta abordagem é a maior flexibilidade e robustez a mudanças na infraestrutura, uma capacidade relevante em cenários da Indústria 4.0, nos quais as linhas de produção são dinâmicas e a reconfiguração de hardware é uma necessidade operacional frequente.

Contudo, a principal desvantagem desta solução é a maior complexidade inicial na modelagem. A criação de um grafo de conhecimento semanticamente correto, que integra múltiplas ontologias, exige significativamente mais esforço de projeto e conhecimento técnico do que a abordagem anterior, que representava a planta por meio de um único documento TD. Essencialmente, esta abordagem transfere a complexidade da lógica de programação do agente para a representação do conhecimento, demandando maior investimento inicial na fase de projeto.

Além disso, a limitação de escopo mais importante da solução é o acoplamento de processo (dependência da sequência de operações na lógica do agente). A arquitetura

proposta reduz o acoplamento nominal no cenário avaliado, porém a lógica do fluxo de trabalho (sequência de funções a serem executadas e a ordem entre elas) permanece explicitamente codificada nos planos do agente. Consequentemente, mudanças radicais no processo industrial, que alterem a ordenação das tarefas, ainda exigiriam uma reprogramação da lógica do agente.

8. Considerações Finais

O presente trabalho partiu do problema do acoplamento nominal em sistemas de automação, uma limitação identificada em uma arquitetura anterior baseada em WoT, na qual o agente de controle dependia de identificadores estáticos para interagir com os dispositivos da planta. A solução proposta consiste em uma arquitetura que introduz uma camada de conhecimento, materializada como um grafo de conhecimento em RDF. A arquitetura foi implementada por meio do framework Apache Jena, utilizando como cliente um agente de software desenvolvido com a plataforma JaCaMo. O principal resultado da avaliação experimental foi a demonstração de que o agente de software é capaz de se adaptar a mudanças na infraestrutura de hardware, como a substituição de um atuador, sem qualquer modificação em seu código-fonte.

Os resultados obtidos e as limitações identificadas apontam para diversas direções promissoras de trabalhos futuros. A sugestão mais imediata é a aplicação da metodologia de modelagem nas outras bancadas da linha de produção FESTO, o que permitiria validar a escalabilidade da arquitetura e explorar o raciocínio do agente em um fluxo de produção mais amplo e complexo.

Para superar a limitação do acoplamento de processo, uma linha de pesquisa promissora seria a modelagem da própria lógica sequencial no grafo de conhecimento. Tal representação permitiria que a sequência de operações se tornasse uma fonte de conhecimento consultável, desacoplando-a da lógica interna dos planos do agente e possibilitando o desenvolvimento de sistemas de controle mais flexíveis e reconfiguráveis.

Finalmente, a utilidade da arquitetura pode ser explorada em cenários que envolvam maior heterogeneidade tecnológica, um desafio comum em ambientes industriais dinâmicos. Outra possibilidade consiste em investigar a integração transparente de dispositivos IoT nativos, que expõem suas próprias APIs. Ao descrevê-los semanticamente e adicioná-los ao grafo, seria possível avaliar a capacidade do agente de orquestrar um sistema híbrido, composto por componentes legados (controlados via OPC UA) e dispositivos modernos, utilizando a mesma lógica de descoberta funcional.

Referências

Beckett, D. and Berners-Lee, T. (2011). Turtle-Terse RDF Triple Language. *https://www.w3.org/TeamSubmission/turtle/*.

Berners-Lee, T., Hendler, J., and Lassila, O. (2023). The semantic web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities. In *Linking the world's information: essays on Tim Berners-Lee's invention of the World Wide Web*, pages 91–103.

Buchgeher, G., Gabauer, D., Martinez-Gil, J., and Ehrlinger, L. (2021). Knowledge graphs in manufacturing and production: a systematic literature review. *IEEE Access*, 9:55537–55554.

Charpenay, V., Zimmermann, A., Lefrançois, M., and Boissier, O. (2022). Hypermedea: a framework for web (of things) agents. In *Companion Proceedings of the Web Conference 2022*, pages 176–179.

Cyganiak, R., Wood, D., and Lanthaler, M. (2014). RDF 1.1 concepts and abstract syntax. W3C Recommendation, World Wide Web Consortium (W3C).

Hermann, M., Pentek, T., and Otto, B. (2016). Design principles for industrie 4.0 scenarios. In *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pages 3928–3937.

Ji, S., Pan, S., Cambria, E., Marttinen, P., and Yu, P. S. (2021). A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE transactions on neural networks and learning systems*, 33(2):494–514.

Kaebisch, S., McCool, M., and Korkan, E. (2023). Web of Things (WoT) Thing Description 1.1. W3C Recommendation, World Wide Web Consortium (W3C).

Kagermann, H., Helbig, J., Wahlster, W., and Hellinger, A. (2013). *Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0: Securing the Future of German Manufacturing Industry ; Final Report of the Industrie 4.0 Working Group*. Forschungsunion.

Klyne, G. (2004). Resource Description Framework (RDF): Concepts and abstract syntax. *http://www.w3.org/TR/rdf-concepts/*.

Leitão, P. (2009). Agent-based distributed manufacturing control: A state-of-the-art survey. *Engineering applications of artificial intelligence*, 22(7):979–991.

Leitão, P., Karnouskos, S., Ribeiro, L., Lee, J., Strasser, T., and Colombo, A. W. (2016). Smart agents in industrial cyber–physical systems. *Proceedings of the IEEE*, 104(5):1086–1101.

Monostori, L., Kádár, B., Bauernhansl, T., Kondoh, S., Kumara, S., Reinhart, G., Sauer, O., Schuh, G., Sihn, W., and Ueda, K. (2016). Cyber-physical systems in manufacturing. *CIRP Annals*, 65(2):621–641.

Nardin, L. G. (2024). 2024 Summer School on AI for Industry 4.0: Summer School on AI technologies for trust, interoperability, autonomy and resilience in industry. Disponível em: `https://ci.mines-stetienne.fr/ai4industry/2024/`. Acesso em: June 2026.

Szilagyi, I. and Wira, P. (2016). Ontologies and Semantic Web for the Internet of Things - a survey. In *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 6949–6954. IEEE.

Zou, Y., Finin, T., Ding, L., Chen, H., and Pan, R. (2003). Using Semantic Web Technology in Multi-Agent Systems: a case study in the TAGA Trading agent environment. In *Proceedings of the 5th international conference on Electronic commerce*, pages 95–101.