A Comparative Analysis of Web Environment Representation for Agent Adaptation

Iderli P. Souza Filho [1], **Jomi F. Hübner**[1]

[1] Pós-Graduação em Engenharia de Automação e Sistemas –
Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brasil

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

Abstract. *This paper presents the results of a qualitative comparative analysis between two standardized approaches for describing Web environments: Thing Description (TD), proposed by the W3C for the Web of Things, and the Resource Description Framework (RDF), an ontology-based approach. The central motivation of this work lies in the need to equip agents operating on the Web with mechanisms that allow them to autonomously and adaptively discover, interpret, and interact with Web resources, even in the face of environmental diversity and variability. To achieve this, it is essential that the environment is described in a formal and interpretable manner, enabling decoupling between agents and environment, in other words, allowing agents to discover their environment at runtime, without relying on specific or pre-configured settings. Based on this comparison, the paper discusses how the choice of a description approach influences the agents' ability to discover and analyze resources in an adaptive and efficient manner in Web environments. The analysis paves the way for future investigations that explore hybrid approaches, reconciling operational efficiency with inference capability and scalability.*

Resumo. *Este artigo apresenta os resultados de uma análise qualitativa comparativa entre duas abordagens padronizadas para descrição de ambientes Web: Thing Description (TD), proposta pela W3C para a Web das Coisas, e o Resource Description Framework (RDF), uma abordagem baseada em ontologias. A motivação central deste trabalho está na necessidade de dotar agentes inseridos na Web de mecanismos que lhes permitam descobrir, interpretar e interagir com recursos Web de forma autônoma e adaptável, mesmo diante da diversidade e variabilidade dos ambientes. Para isso, é importante que o ambiente seja descrito de maneira formal e interpretável, possibilitando o desacoplamento entre agentes e ambiente, ou seja, permitindo que os agentes descubram seu ambiente em tempo de execução e portanto não dependam de configurações específicas ou pré-programadas. A partir dessa comparação, discute-se como a escolha da abordagem de descrição do ambiente influência a capacidade dos agentes de descobrir e analisar recursos de forma adaptável e eficiente em ambientes Web. A análise abre espaço para investigações futuras que explorem abordagens híbridas, conciliando eficiência operacional com capacidade de inferência e escalabilidade.*

1. Introdução

À medida em que se explora os desafios da automação em sistemas distribuídos, torna-se fundamental considerar o papel desempenhado pelos agentes e pelos Sistemas Multiagentes (SMA). Esses desafios se tornam ainda mais evidentes quando se trata da integração de agentes em ambientes *Web*, onde é necessário que os agentes sejam capazes de se adaptar a uma ampla gama de cenários variáveis, mantendo a eficiência na realização de seus objetivos.

Nesse contexto, Ciortea et al. [Ciortea et al. 2018] argumentam que a Web das Coisas (WoT) oferece novas oportunidades para a pesquisa em agentes, destacando a importância de adaptar os princípios da *Web* para apoiar interações entre sistemas distribuídos. Complementarmente, Boissier [Boissier et al. 2021] observa que os recentes esforços de padronização da Web das Coisas e do *Linked Data* estão transformando a hipermídia em uma estrutura de informação homogênea e interconectada, possibilitando que agentes operem de forma mais eficaz sobre esse tecido hipermidiático.

Este trabalho busca compreender como diferentes formatos de representação de recursos na *Web* influenciam a capacidade de agentes atuarem de maneira eficiente e flexível. Em ambientes *Web* altamente dinâmicos, a forma como os recursos são descritos, estruturados e disponibilizados afeta diretamente a habilidade dos agentes de descobri-los, interpretá-los e interagir com eles. Assim, o estudo investiga como diferentes modelos de representação, como *Thing Descriptions* (TDs) e *Resource Description Framework* (RDF), suportam um fraco acoplamento entre agentes e ambiente facilitando a adaptabilidade a mudanças.

Comparar diferentes abordagens de acesso a ambientes *Web* passa pela avaliação de como cada abordagem trata aspectos fundamentais, como a forma de descrição dos recursos disponíveis no ambiente, a facilidade de acesso e interpretação por parte dos agentes, o suporte à inferência lógica e o nível de complexidade de implementação. A partir dessa análise, é possível identificar as abordagens que favorecem um melhor uso do agente e alinhar um maior desacoplamento entre os agentes e o ambiente, facilitando sua adaptação a diferentes configurações e mudanças no ambiente.

As próximas seções deste artigo estão assim organizadas: a Seção 2 aborda os fundamentos teóricos, incluindo Agentes na *Web*, Web Semântica, *TD* e *RDF*; a Seção 3 discute trabalhos relacionados, contextualizando a pesquisa; a Seção 4 detalha a metodologia utilizada; a Seção 5 compara as abordagens sem descrição, *TD* e *RDF*, analisando com base em critérios descritos; e, finalmente, a Seção 6 apresenta as conclusões e sugere pesquisas futuras.

2. Fundamentação Teórica

Esta seção apresenta os conceitos essenciais que embasam a pesquisa. Primeiro, discutimos os Agentes *Web*, capazes de operar autonomamente em ambientes *Web*. Em seguida, exploramos a *Web Semântica*, que permite a interpretação automatizada de dados na *Web*. Por fim, detalhamos duas abordagens-chave para descrição de recursos: *Thing Description*, padrão da Web das Coisas, e *Resource Description Framework*, baseado em ontologias.

2.1. Agentes Web

Agentes *Web*, como definidos por Boudriga [Boudriga and Obaidat 2004], são agentes capazes de acessar recursos remotos e, com isso, adquirir conhecimento sobre a infraestrutura da rede e os serviços disponíveis. O principal diferencial de um ambiente *Web* para agentes está em sua estrutura dinâmica e na vasta quantidade de informação com a qual o agente precisa lidar. Esses ambientes são extremamente voláteis, com mudanças frequentes na disponibilidade de recursos, na organização das informações e nas condições da rede. Assim, o agente deve ser capaz de lidar com essa dinamicidade, adaptando suas estratégias de busca e atuação para continuar eficaz em sua missão. Nesse contexto, a forma como os dados são apresentados ao agente desempenha um papel fundamental na sua capacidade de operar de maneira autônoma e eficiente. Dados bem estruturados, com descrições semânticas padronizadas e acessíveis, facilitam a descoberta, interpretação e integração da informação. Assim, para que agentes *Web* possam lidar com ambientes *Web*, é essencial que os recursos sejam representados de forma compreensível por máquina, com interoperabilidade e com suporte a adaptação.

2.2. Web Semântica

Segundo Berners-Lee, em [Berners-Lee et al. 2023], a Web Semântica é uma extensão da *Web* atual na qual as informações possuem significados bem definidos, permitindo que computadores e pessoas colaborem de forma mais eficaz. O objetivo é criar um ambiente onde agentes autônomos, ao navegar entre páginas, possam realizar tarefas complexas em nome dos usuários.

Para que essa visão seja concretizada, os computadores devem ter acesso a coleções estruturadas de dados, bem como a conjuntos de regras de inferência que lhes permitam realizar raciocínios automatizados. Nesse contexto, a Web Semântica promove um modelo avançado de representação de conhecimento, no qual a informação é organizada de maneira lógica e padronizada, facilitando a compreensão, o compartilhamento e a reutilização do conhecimento entre diferentes agentes e sistemas.

2.3. Thing Description

A *Web of Things* (*WoT*) visa integrar objetos físicos do mundo real à infraestrutura da *World Wide Web*. Na arquitetura proposta pela W3C, as camadas iniciais são compostas por *Thing Descriptions* (*TDs*), que representam metadados padronizados descrevendo as capacidades dos dispositivos *WoT* de forma uniforme. Essas descrições fornecem um conjunto de interações com base em um vocabulário comum, permitindo a integração de dispositivos heterogêneos e possibilitando a interoperabilidade entre diferentes aplicações e sistemas [Consortium et al. 2023]. O objetivo principal das *TDs* é descrever os chamados "*Things*" (coisas/dispositivos físicos) de modo que sistemas e agentes de software possam descobri-los e interagir com eles de maneira automatizada.

Com o uso de *TD*, os agentes passam a dispor de um modelo descritivo padronizado que explicita, de forma estruturada, as possíveis interações com dispositivos ou serviços disponíveis na *Web*. Esse modelo organiza as capacidades dos recursos em três categorias principais: propriedades observáveis, ações executáveis e eventos monitoráveis. Tais interações são representadas por meio do conceito de *affordances*. A teoria de *affordance* conecta a prática com a percepção, as *affordances* de um objeto ou

ambiente são as possibilidades de ação que ele invoca para um sujeito que o percebe [Fayard and Weeks 2014]. Assim, ao identificar *affordances* nos recursos *Web*, o agente é capaz de inferir automaticamente quais operações pode realizar, como consultar o estado de uma máquina, ativar um processo ou assinar notificações de eventos. Esse mecanismo não apenas promove maior autonomia, como também favorece a reutilização e adaptação dos agentes a novos contextos e dispositivos, desde que estes estejam descritos conforme o padrão *TD*. A formalização dessas *affordances* transforma a *Web* em um espaço de possibilidades concretas de ação, permitindo que agentes interajam com o ambiente de forma dinâmica, interpretando as funcionalidades dos recursos com base nas informações oferecidas pela descrição.

2.4. Resource Description Framework (RDF)

O *Resource Description Framework* (*RDF*) é uma forma de descrição de metadados para expressar informações sobre recursos. Recursos podem ser qualquer coisa, incluindo documentos, pessoas, objetos físicos e conceitos abstratos [Consortium et al. 2014]. Essa estrutura destina-se a situações em que as informações na *Web* precisam ser processadas por aplicativos, em vez de serem exibidas apenas para pessoas. Em particular, o *RDF* pode ser usado para publicar e interligar dados na *Web*. O *RDF* permite representar informações na forma de triplas (sujeito, predicado e objeto). Essa estrutura promove conexões semânticas entre os elementos, revelando relações de contexto e hierarquias. Através de inferência lógica, o agente pode deduzir novos conhecimentos a partir de dados existentes, possibilitando ao agente interpretar essas declarações como crenças sobre o estado do ambiente

O *RDF* é um padrão que possibilita a codificação, troca e reutilização de metadados estruturados [Miller 1998], isto pode promover uma melhor interoperabilidade entre diferentes fontes de informação. De acordo com Candan [Candan et al. 2001], o *RDF* permite que agentes de software encontrem, filtrem e integrem dados de forma mais inteligente, graças à sua representação orientada a significados e relações. Diferentemente de modelos mais rígidos, o *RDF* não apenas descreve funcionalidades, mas também expressa relações entre entidades, categorias hierárquicas e propriedades semânticas de dispositivos e serviços. Essa característica confere aos agentes a habilidade de raciocinar sobre os elementos do ambiente que estão acessando, essa característica é ainda mais observável quando complementada com ontologias. Por exemplo, um agente pode deduzir que um novo dispositivo pertence a uma determinada classe funcional já conhecida, ou inferir a presença de funcionalidades a partir de heranças semânticas definidas na ontologia.

3. Trabalhos Relacionados

Nesta seção, buscamos identificar trabalhos que abordam comparações entre abordagens voltadas à representação de páginas e recursos *Web* de forma compreensível por agentes de software. O trabalho de Szilagyi [Szilagyi and Wira 2016] apresenta uma visão geral sobre o uso de abordagens da *Web Semântica* e ontologias no contexto da *Internet of Things* (*Iot*), com ênfase na padronização da descrição de dispositivos e dos dados que produzem. O estudo se relaciona com este trabalho ao destacar a importância de representações semânticas que permitam o processamento automático da informação, tornando os recursos mais acessíveis a sistemas computacionais. Ambos os trabalhos reconhecem que a forma como os dados são estruturados influencia diretamente a capacidade

de sistemas inteligentes interpretarem e atuarem sobre o ambiente digital. Entretanto, o foco do artigo de Szilagyi está mais centrado na organização e integração de dados no domínio da *IoT*, enquanto este trabalho direciona sua análise à forma como diferentes abordagens de descrição influenciam a capacidade de ação de agentes na *Web*. Ou seja, aqui o interesse está não apenas na representação em si, mas em como ela afeta diretamente o comportamento e a tomada de decisão de agentes que operam em ambientes distribuídos e dinâmicos.

Outro estudo que aborda a integração de abordagens da *Web Semântica* com sistemas *multiagentes* é o de Zou et al. [Zou et al. 2003]. Neste trabalho, os autores demonstram como linguagens baseadas em ontologias, como *RDF* e *OWL*, podem ser incorporadas a sistemas multiagentes baseados nos padrões da *FIPA* (*Foundation for Intelligent Physical Agents*). Essa integração visa criar ambientes de simulação mais flexíveis e realistas, especialmente voltados para mercados dinâmicos e abertos, onde agentes autônomos precisam negociar e cooperar com base em conhecimento semântico compartilhado. Entretanto, o estudo não tem como foco a comparação entre diferentes abordagens semânticas, tampouco realiza uma análise crítica dos ganhos e limitações que tais linguagens trazem para os sistemas multiagentes. O diferencial do presente trabalho está justamente na análise comparativa focada nos formatos de descrição de páginas e recursos *Web*, considerando não apenas a representação dos dados, mas também o impacto direto dessas abordagens na capacidade dos agentes de descobrir, interpretar e interagir de forma adaptável e eficiente com ambientes *Web*.

4. Metodologia de Pesquisa

Durante o hackathon do evento *AI4Industry 2024* [Nardin 2024], foi proposto um desafio que explorou a integração de tecnologias como *Wot*, *Knowledge Graphs* e Agentes Autônomos, combinadas com os princípios de Inteligência Artificial Confiável e Responsável, em um caso de uso industrial de fácil compreensão. O objetivo central foi permitir que agentes autônomos coordenassem e regulassem diferentes máquinas dentro de uma fábrica, a fim de controlar o processo de fabricação de produtos. As máquinas da fábrica foram disponibilizadas como serviços *Web* padronizados, utilizando *TD* e *RDF* como formas de descrição do ambiente na *Web*.

Esse cenário motivou a elaboração deste artigo, no qual, para avaliar a efetividade de diferentes abordagens de descrição de ambientes, implementamos três versões distintas de um agente responsável por acessar e interagir com o ambiente proposto no AI4Industry. A primeira versão opera sem nenhuma descrição formal do ambiente, com o objetivo de estabelecer uma *baseline*, representando um cenário comum em que o agente acessa diretamente os links do ambiente. As demais versões utilizam descrições formais: a segunda versão adota *TD*, enquanto a terceira é baseada em *RDF*.

As três implementações consideram uma simulação de fábrica que recebe recipientes (como potes de iogurte) e estoques de insumos (iogurte) de fornecedores. Esses recipientes são então preenchidos e embalados na própria fábrica para posterior comercialização no varejo. Em cada máquina dessa fábrica, há diferentes componentes, como autômatos, máquinas e outros dispositivos que processam informações captadas por sensores (como sensores de presença, temperatura, umidade, peso, etc.) e enviam comandos a atuadores (como braços robóticos, cilindros, entre outros) para manipular os

materiais. Cada máquina possui seu próprio estado interno, descrito por diversas variáveis que representam o status dos sensores e atuadores, o estado de funcionamento dos processos (em operação, com erro, aguardando intervenção, etc.), além dos dados recebidos e enviados. É importante destacar que as máquinas são independentes entre si, com formatos próprios de dados, estratégias de decisão e métodos de gerenciamento.

Para comparar não apenas a capacidade de representação de cada abordagem, mas também suas implicações práticas, delimitamos cinco critérios para avaliar as três implementações. Os critérios selecionados visam destacar aspectos fundamentais que influenciam diretamente a atuação de agentes em ambientes *Web*:

- (I) **O que o agente sabe previamente:** Determina os dados necessários que o agente deve possuir para descobrir as informações necessárias para atuar no ambiente.
- (II) **Descrição do ambiente**: Revela os conceitos utilizados para descrever o ambiente.
- (III) **Adaptação a mudanças:** Avalia o nível de mudança que pode acontecer no ambiente sem que o agente precise de alterações na sua implementação.
- (IV) **Suporte à inferência lógica:** Determina o potencial das abordagens em promover raciocínios automatizados por parte do agente.
- (V) **Complexidade de implementação:** Apresenta o que é necessário que um desenvolvedor entenda para aplicar a abordagem.

Com relação às limitações de nosso escopo, reconhecemos que agentes autônomos poderiam acessar e interpretar um ambiente *Web* utilizando outras abordagens, como *Schema.org* [Patel-Schneider 2014], *OWL* [Antoniou and Harmelen 2009] e *OWL 2 RL* [Hitzler et al. 2009]. No entanto, optamos por focar a comparação em *TD* e *RDF*, pois ambas as abordagens são recomendadas e ativamente promovidas pelo *W3C* em iniciativas relacionadas à Web das Coisas e à Web Semântica.

5. Comparação de diferentes representações do ambiente

Para detalhar o funcionamento das três implementações usadas na comparação, apresentamos três algoritmos que ilustram o acesso de um agente ao ambiente: sem uso de descrição (1), com *TD* (2) e com *RDF* (3).

Algorithm 1 GetPropertyDirect from operation URL

```
1: function GETPROPERTYDIRECT(operationURL)
2:     Send HTTP GET request to operationURL
3:     Parse JSON response to get property value value
4:     return value
5: end function
```

O algoritmo (1), representa uma abordagem sem descrição do ambiente para obter o valor de uma propriedade de um dispositivo da planta. A função recebe como entrada a URL da operação que retorna a propriedade desejada. Ela envia uma requisição *HTTP GET* para essa *URL*, recebe a resposta no formato *JSON*, extrai o valor da propriedade a partir dessa resposta e o retorna ao solicitante. Esse comportamento evidencia que o agente precisa conhecer previamente a URL da operação para obter o valor da propriedade. Caso a planta altere essa URL, o agente deixará de funcionar, demonstrando o alto acoplamento entre agente e ambiente.

Algorithm 2 GetPropertyTD from *storageRack* Thing Description

```
1: function GETPROPERTYTD(URL_TD_storageRack, propertyName)
2:     Load TD from URL_TD_storageRack
3:     Find property propertyName in TD
4:     if property propertyName has a Form form with Target URI URI then
5:         Prepare form preparedForm from form
6:         Send HTTP GET request to URI using preparedForm
7:         Parse JSON response to get property value value
8:         return value
9:     else
10:        return null                                  ▷ property or form not found
11:    end if
12: end function
```

O algoritmo 2, detalha uma abordagem para acessar uma propriedade de um dispositivo utilizando a *TD*. A função recebe a *URL* da *TD* do dispositivo e o nome da propriedade desejada. Primeiro, a *TD* é carregada a partir da *URL* fornecida. Em seguida, busca-se na *TD* a propriedade especificada e se ela possui um *form* com uma *URI* de destino para acesso. Se esse *form* existir, um formulário é preparado para a requisição, que é então enviada via *HTTP GET* para a *URI* alvo. A resposta *JSON* é processada para extrair o valor da propriedade, que é retornado ao solicitante. Caso a propriedade ou o formulário não sejam encontrados na *TD*, a função retorna *null*, indicando que o acesso não foi possível.

Nesta abordagem, o agente não precisa conhecer previamente a *URL* da operação, pois ela é descoberta a partir do *TD*. Isso reduz o acoplamento entre agente e ambiente, permitindo que mudanças na planta, como a alteração de uma *URL*, não sejam refletidas no agente. Assim, o agente continua funcionando normalmente, desde que o nome da propriedade permaneça o mesmo.

Algorithm 3 GetPropertyRDF from *storageRack* RDF and Ontology

```
1: function GETPROPERTYRDF(URL_RDF_storageRack, URL_Ontology, classOperation)
2:     Load ontology from URL_Ontology
3:     Crawl RDF graph from URL_RDF_storageRack
4:     for all Subject thing of type Thing in RDF graph do
5:         if thing is of type classOperationURI according to ontology then
6:             for all Affordance af of thing do
7:                 if af is of type property affordance as defined in ontology then
8:                     if af has a Form form with Target URI then
9:                         Prepare form preparedForm from form
10:                        Send HTTP GET request to Target URI using preparedForm
11:                        Parse JSON response to extract property value value
12:                        return value
13:                    end if
14:                end if
15:            end for
16:        end if
17:    end for
18:    return null                                      ▷ if property not found
19: end function
```

Por último, o algoritmo (3), descreve um método para acessar uma propriedade de

um dispositivo representado em *RDF*, utilizando uma ontologia para filtrar o tipo correto da coisa (*Thing*).É importante destacar que, para seguir o funcionamento do algoritmo, o documento em *RDF* deve ser estruturado utilizando a ontologia de *TD* da W3C, incorporando assim as características de *Thing* e *affordance*, o que facilita a comparação entre as abordagens. A função recebe três parâmetros: a *URL* da descrição *RDF* do dispositivo (*URL_RDF_storageRack*), a URL da ontologia (*URL_Ontology*) e a URI da classe da operação desejada (*classOperation*). Primeiramente, a ontologia é carregada para permitir que o sistema entenda e compare semanticamente os tipos de elementos encontrados no *RDF*. Em seguida, o grafo *RDF* do dispositivo é analisado e todos os recursos do tipo "*Thing*" são listados. Para cada um desses recursos, o algoritmo verifica se ele é do tipo correspondente à classe da operação definida na ontologia. Caso seja, o sistema percorre suas *affordances* e seleciona aquelas que são do tipo "propriedade", conforme definido na ontologia. Se uma dessas *affordances* possuir um formulário com uma *URI* de destino, o algoritmo prepara esse formulário (caso precise incluir cabeçalhos, parâmetros, etc.), envia uma requisição *HTTP GET* para a *URI* e, ao receber a resposta no formato *JSON*, extrai e retorna o valor da propriedade. Se nenhuma correspondência válida for encontrada ao longo do processo, a função retorna *null*.

Na abordagem com *RDF*, o agente também não precisa conhecer previamente a *URL* da operação, pois ela é descoberta dinamicamente a partir do grafo *RDF*. No entanto, diferentemente do *TD*, o vínculo entre agente e ambiente é estabelecido com base na classe da operação, definida pela ontologia, e não pelo nome ou identificador direto da propriedade.

Com base no cenário prático desenvolvido, a Tabela 1 resume os resultados da comparação considerando os critérios definidos. As próximas seções detalham essas avaliações.

Tabela 1. Comparação das abordagens de descrição de ambiente *Web*.

Critério	**Sem descrição** (1)	**Thing Description** (2)	**RDF** (3)
(I)	Link direto da operação	Link do TD e Nome da Operação no TD	Link da Ontologia, Link do RDF e Classe da Operação.
(II)	URL da Operação	Affordances: ações, propriedades, eventos e interações dos dispositivos	Classes, propriedades, funções e relações semânticas dos dispositivos
(III)	A URL da operação deve ser mantida	Permite mudanças mantendo o nome do TD	Altamente adaptável via evolução da ontologia, desde que mantenha a classe da operação.
(IV)	Nenhum	Exige motor externo	Permite herança e raciocínio baseado em ontologias
(V)	Requer conhecimento de HTTP	Requer conhecimento de TD	Requer conhecimento de RDF e de Ontologia

5.1. Abordagem Sem Descrição

A avaliação de um agente acessando um ambiente por meio de links diretos, sem uma descrição, apresentou estes resultados:

Critério (I): O acesso ocorre por meio de *URIs* fixas definidas pelo projetista do ambiente.

Critério (II): Pela ausência de uma descrição formal, a única coisa que o agente consegue acessar do ambiente é o resultado da operação que será realizada.

Critério (III): Sem descrição do ambiente o agente não é adaptável. Como o agente só visualiza de forma direta o link da operação, mudanças ou evoluções do ambiente, que impliquem na mudança da *URL*, resultam na perda de funcionamento do agente. Essa rigidez torna o sistema frágil e pouco adaptável em ambientes que passam por mudanças regulares.

Desta forma, o acoplamento entre o agente e o ambiente é alto. O agente depende de estruturas fixas e específicas previamente conhecidas. Como não há uma descrição do ambiente que forneça as funcionalidades e operações, o agente não consegue descobrir ou utilizar novas funcionalidades.

Critério (IV): Não há suporte à inferência lógica, por parte da dimensão do ambiente, uma vez que não há um documento base estruturado a partir do qual seja possível extrair ou relacionar conceitos, metadados ou vocabulários definidos.

Critério (V): Por outro lado, existe uma simplicidade na implementação do agente.

5.2. Abordagem Utilizando Thing Description

A abordagem de acesso por meio de *TD* apresentou os seguintes resultados:

Critério (I): Para acessar o ambiente descrito via *TD*, o agente precisa de dois elementos principais: (1) a URL do documento *TD* e (2) o nome da propriedade ou operação desejada. A partir da URL, o agente carrega o *TD*, localiza a *property* correspondente ao nome fornecido e, em seguida, identifica a *form* apropriada, contendo o *endpoint* (URI) e o método de acesso (por exemplo, HTTP *GET*). Com isso, o agente prepara e envia a requisição para recuperar o valor desejado.

Critério (II): Nesta abordagem, a partir do documento *TD*, o agente identifica e lista as *affordances* disponíveis no ambiente. Essas *affordances*, chamadas formalmente de *InteractionAffordances*, representam os diferentes tipos de operações que uma *Thing* pode oferecer para que usuários ou sistemas externos possam controlar ou obter informações sobre ela. Cada *affordance* está associada a um ou mais *forms*, que descrevem como a operação pode ser realizada, especificando protocolos e métodos compatíveis (como HTTP, CoAP, MQTT).

Essas *affordances* englobam ações, propriedades, eventos, permitindo ao agente compreender como acessar e controlar o dispositivo. Assim, o agente pode ler ou modificar estados através das propriedades, executar comandos por meio das ações e receber notificações assíncronas ao se inscrever em eventos.

Critério (III): Mudanças internas na estrutura do dispositivo ou serviço descrito podem ser absorvidas desde que o nome e o link do *TD* permaneçam os mesmos. Como o agente interage com o ambiente por meio de sua descrição (o *TD*), há um desacoplamento entre o agente e o dispositivo. O agente não precisa conhecer diretamente os detalhes internos do dispositivo, ele depende apenas da estrutura definida no *TD*, que funciona

como uma interface estável. Assim, desde que o *TD* continue válido, o agente pode continuar operando mesmo que o dispositivo sofra alterações.

Critério (IV): Embora o *TD* ofereça uma estrutura padronizada para descrever dispositivos e seus pontos de interação, ele não possui mecanismos nativos para inferência lógica. Um agente que utiliza apenas o TD não consegue realizar raciocínios baseados em relações conceituais, hierarquias ou regras. Para isso, seria necessário integrá-lo com motores externos de inferência. Assim, o TD descreve o como interagir, mas não o significado mais amplo dos recursos.

Critério (V): Para utilizar o *TD*, é necessário um conhecimento básico sobre sua estrutura e funcionamento. Contudo, o *TD* apresenta uma estrutura simples e de fácil compreensão, baseada em *JSON-LD*, o que facilita seu aprendizado e adoção por desenvolvedores e integradores. Sua organização, com seções bem definidas para propriedades, ações e eventos, permite uma correlação direta entre os elementos descritos e as funcionalidades reais do dispositivo. Essa simplicidade contribui para uma curva de aprendizado mais suave, especialmente em comparação com abordagens baseadas em ontologias, que exigem familiaridade com lógica descritiva e vocabulários formais.

5.3. Abordagem Utilizando Resource Description Framework

Na abordagem baseada em *RDF*, os resultados apresentados foram os seguintes:

Critério (I): O agente precisa de três informações essenciais para acessar e interpretar corretamente o ambiente descrito em *RDF*: a URL da ontologia utilizada, a URL do *RDF* que descreve o ambiente, e a classe da operação desejada.

A URL da ontologia funciona como um vocabulário conceitual que permite ao agente compreender o significado de propriedades, ações, eventos e as relações hierárquicas entre eles. A URL do *RDF* contém a descrição do ambiente e das coisas disponíveis para interação. Já a classe da operação indica o tipo de funcionalidade que o agente está buscando executar. A necessidade dessas três informações está no fato de que o agente realiza uma comparação entre a classe da operação especificada (o que ele deseja fazer) e as classes de operação descritas no *RDF*, usando a ontologia como referência. Essa comparação permite garantir que a operação buscada está de acordo com o esperado na ontologia e pode ser acessada naquele ambiente.

Critério (II): O ambiente é descrito por meio de um grafo *RDF* que representa as informações e operações de forma detalhada e estruturada. Esse grafo utiliza classes, propriedades, funções e relações semânticas definidas na ontologia, o que permite descrever não apenas as operações e estados dos dispositivos, mas também as relações entre eles, hierarquias, restrições e outros aspectos contextuais. Ao acessar esses dados, o agente pode descobrir as operações disponíveis, compreender seu significado e determinar como executá-las.

Critério (III): A abordagem com *RDF* oferece maior adaptabilidade a mudanças. Através do uso de ontologias, os agentes não dependem de identificadores específicos, mas sim de conceitos e relacionamentos semânticos. Isso permite que, mesmo quando novas operações ou estruturas surgem, o agente consiga reconhecê-las por similaridade conceitual ou herança semântica. Em outras palavras, a evolução da ontologia permite que o agente se ajuste automaticamente a novos contextos, sem necessidade de alterações

manuais.

Critério (IV): A abordagem baseada em *RDF* oferece suporte à inferência lógica, principalmente quando utilizada em conjunto com ontologias *OWL* e motores de raciocínio. Esse suporte permite que agentes derivem novos conhecimentos a partir de fatos existentes no grafo, identifiquem hierarquias de classes, relações implícitas e comportamentos esperados de dispositivos com base em regras formais. Por exemplo, mesmo que um agente não conheça diretamente um determinado tipo de propriedade, ele pode reconhecê-la como válida por meio de herança ontológica ou equivalência semântica.

Critério (V): Um dos aspectos críticos a se considerar ao adotar *RDF* como modelo de descrição semântica para ambientes acessados por agentes autônomos é a curva de aprendizado associada à sua estrutura e uso efetivo. Ao contrário de abordagens mais diretamente operacionais, como *TD*, o *RDF* exige um conhecimento prévio mais profundo tanto em termos de sintaxe quanto de semântica.

Além disso, há uma necessidade de padronização na escrita do *RDF* com relação à ontologia adotada. Isso significa que quem fornece os dados (por exemplo, os desenvolvedores) devem seguir consistentemente os termos, classes e propriedades definidos na ontologia, evitando ambiguidade ou variações livres que possam comprometer a interoperabilidade semântica.

6. Considerações Finais

Este trabalho partiu da motivação de compreender como diferentes formas de representação de recursos na *Web* influenciam a capacidade de agentes autônomos atuarem com menor acoplamento em ambientes distribuídos, particularmente na Web das Coisas (*WoT*). Em cenários cada vez mais dinâmicos, a maneira como dispositivos e serviços são descritos impacta diretamente na autonomia dos agentes e em sua capacidade de adaptação com sistemas heterogêneos.

A comparação entre as abordagens, sem descrição, *TD* e *RDF*, evidencia diferentes níveis de acoplamento, expressividade e adaptabilidade para agentes acessando ambientes *Web*. A abordagem sem descrição mostrou maiores limitações, o agente depende de URIs fixas e não possui meios para descobrir ou interpretar funcionalidades do ambiente de forma dinâmica. Em contraste, tanto *TD* quanto *RDF* introduzem mecanismos de descrição que permitem um maior desacoplamento e autonomia do agente.

As abordagens *TD* e *RDF* trazem pontos distintos para agentes ao descreverem o ambiente *Web* de forma estruturada. O *TD* permite que o agente descubra e acesse funcionalidades a partir de um documento único, por meio da descrição das affordances. Já o *RDF*, ao utilizar ontologias, permite que o agente interprete relações e conceitos, possibilitando uma maior capacidade de inferência lógica e adaptação a mudanças. Entretanto, o algoritmo apresentado demonstrou que, para que um agente utilize a abordagem *RDF*, é necessário que ele possua mais informações prévias. Essa necessidade implica em um aumento da complexidade no desenvolvimento do agente, exigindo um conhecimento técnico mais aprofundado sobre modelos semânticos e ontologias.

Como conclusão, a escolha entre *TD* e *RDF* depende fortemente do cenário de aplicação: sistemas estáticos e previsíveis se beneficiam da simplicidade do *TD*, enquanto sistemas dinâmicos e complexos encontram maior valor no uso de *RDF* e on-

tologias. Trabalhos futuros podem incluir uma análise quantitativa de desempenho entre as duas abordagens, testes em ambientes reais de produção industrial, e a exploração de métodos híbridos que combinem a padronização operacional do *TD* com a expressividade semântica do *RDF*, potencializando o melhor de ambos os mundos.

Referências

Antoniou, G. and Harmelen, F. v. (2009). Web ontology language: Owl. *Handbook on ontologies*, pages 91–110.

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. Association for Computing Machinery, New York, NY, United States.

Boissier, O., Ciortea, A., Harth, A., and Ricci, A. (2021). Autonomous agents on the web. In *Dagstuhl-Seminar 21072: Autonomous Agents on the Web*, page 100p.

Boudriga, N. and Obaidat, M. S. (2004). Intelligent agents on the web: A review. *Computing in Science & Engineering*, 6(4):35–42.

Candan, K. S., Liu, H., and Suvarna, R. (2001). Resource description framework: metadata and its applications. *Acm Sigkdd Explorations Newsletter*, 3(1):6–19.

Ciortea, A., Mayer, S., and Michahelles, F. (2018). Repurposing manufacturing lines on the fly with multi-agent systems for the web of things. In *Proceedings of the 17th international conference on autonomous agents and multiagent systems*, pages 813–822.

Consortium, W. W. W. et al. (2014). Rdf 1.1 primer.

Consortium, W. W. W. et al. (2023). Web of things (wot) thing description 1.1.

Fayard, A.-L. and Weeks, J. (2014). Affordances for practice. *Information and Organization*, 24(4):236–249.

Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P. F., Rudolph, S., et al. (2009). Owl 2 web ontology language primer. *W3C recommendation*, 27(1):123.

Miller, E. (1998). An introduction to the resource description framework. *D-lib Magazine*.

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/`.

Patel-Schneider, P. F. (2014). Analyzing schema. org. In *The Semantic Web–ISWC 2014: 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I 13*, pages 261–276. Springer.

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.

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.