UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

RODRIGO ZANELLA RIBEIRO

# Proposta de agente blackboard para a resolução da interoperabilidade semântica em IoT

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Rosa Maria Vicari

Porto Alegre
2016

## AGRADECIMENTOS

Durante toda a graduação se vive momentos de alegrias e decepções, em cada momento, tive a felicidade de compartilhar com pessoas excelentes e que se não fossem por elas, esses momentos não seriam tão especiais.

Primeiramente Deus que permitiu que tudo acontecesse, não somente nestes anos como universitário, mas ao longo de minha vida. Também a Universidade Federal do Rio Grande do Sul, seu corpo docente, direção e administração que me deram a oportunidade de aproveitar muito bem esses cinco anos de estudo em uma universidade com confiança e méritos reconhecidos internacionalmente. Em especial, gostaria de agradecer a minha orientadora Dra. Rosa Maria Viccari, pelo suporte desde o tempo de bolsista de iniciação científica até este trabalho de conclusão, pelas correções e incentivos. Agradeço também ao Tiago Primo, que graças ao seu apoio e dedicação me ensinou muito durante meu período como bolsista de iniciação científica.

Agradeço ao pessoal da Cognitiva Brasil, que me acolheu de braços abertos, e me ajudou bastante durante o desenvolvimento do trabalho, com ideias, motivação e explicações dos fundamentos desse trabalho, especialmente sobre sistemas multiagentes.

Agradeço também aos meus amigos do CIC Gourmet, pelas diversas discussões, almoços e ajuda nos estudos dos conteúdos vistos durante a faculdade, especialmente a Dra. Carine Beatrici que me ajudou com a sua experiência para melhorar esse trabalho, ao Jorge W. Filho e ao Bruno Grisci, ambos colegas que se demonstraram muito competentes, agradeço pela amizade e companheirismo durante a graduação.

Agradeço ao meu pai, minha mãe e minha irmã, por sempre acreditarem em mim, e pela vontade, muitas vezes superando a minha, de ter um filho e um irmão estudando em uma das melhores universidades do Brasil, fica meu reconhecimento e agradecimento por muitas vezes ter abdicado de suas aspirações para dar mais oportunidades a mim. Agradeço a minha companheira, amiga, inteligente (principalmente por ter me escolhido), linda, namorada, Jessika Renally Ribeiro Rodrigues, que apesar de distante, sempre esteve muito presente me acalmando, me apoiando e durante o desenvolvimento desse trabalho foi mais importante do que imagina.

A todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

## RESUMO

A grande diversidade de aplicações em Internet das coisas gera um problema de interoperabilidade semântica, na medida que dois dispositivos que pertencem a aplicações diferentes precisam se comunicar. Esse trabalho propõe o agente tradutor, que funciona como um intermediário entre a comunicação entre dois dispositivos IoT. Ele é baseado na arquitetura blackboard dividido em áreas, onde cada área corresponde a um grupo de dispositivos IoT que utilizam a mesma ontologia. Outro agente proposto é o agente de alinhamento, que utiliza o algoritmo de similaridade N-GRAMA e módulos de ontologias dos dispositivos IoT, buscam encontrar conceitos relacionados entre aplicações, de forma que facilite o tratamento do problema de interoperabilidade semântica em Internet das Coisas.

Esses agentes são aplicados em um sistema de impressão. O sistema de impressão é composto pelo agente usuário, que controla uma fila de impressão virtual, interagindo com o sistema para conseguir uma impressora para o usuário. O agente servidor é o outro agente que compõe o sistema e realiza o controle de acesso às impressoras pelo usuário e detém informações básicas do funcionamento delas. O agente impressora controla e interage com as impressoras do sistema. Cada agente do sistema tem sua própria ontologia e o agente tradutor e de alinhamento fazem a conversão das mensagens entre esses agentes.



**Palavras-chave:** Sistemas Multiagentes. Internet das coisas. Interoperabilidade Semântica. Ontologias.

## ABSTRACT

The wide variety of applications in the Internet of Things brings up a semantic interoperability problem, as two devices belonging to different applications need to communicate. This paper proposes the translator agent, which acts as an intermediary between the communication of two IoT devices. It is based on blackboard architecture divided into areas, where each area represents a group of IoT devices using the same ontology. Another proposed agent is the alignment agent, which uses the N-GRAM similarity algorithm and modules of IoT devices ontologies, seek to find related concepts between applications in order to facilitate the treatment of the problem of semantic interoperability in the Internet of Things.

These agents are applied in a printing system. The printing system is composed of the user agent, which controls a virtual print queue, interacting with the system to get a printer to the user. The server agent is another agent that composes the system and performs the user printers to access control and holds basic information about the operation of them. The printer agent controls and interacts with the system printers. Each system has its own ontology agent and the translator and alignment agents make the conversion of messages between these agents.



**Keywords:** Multiagent systems, Internet of things, Semantic Interoperability, Ontologies.

## LISTA DE FIGURAS

Figura 2.1 A convergência das três perspectivas resultam no conceito de Internet das Coisas ..... 16
Figura 2.2 Organização da arquitetura SOA ..... 16
Figura 2.3 Classificação da interoperabilidade em IoT,segundo a IERC ..... 17
Figura 2.4 Percepção e ação no agente ..... 19
Figura 2.5 Cooperação e competição entre agentes ..... 23
Figura 2.6 Protocolo Rede de Contrato ..... 26
Figura 2.7 Processo de matching de ontologias ..... 31

Figura 3.1 User assisted semantic interoperability in Internet of Things ..... 35
Figura 3.2 Módulos de ontologias propostos pelos autores ..... 35

Figura 4.1 Conjunto de ontologias proposto para o trabalho ..... 38
Figura 4.2 Imagem ilustrativa do agente tradutor ..... 39
Figura 4.3 Protocolo de cadastro no agente tradutor ..... 40
Figura 4.4 Protocolo de envio de mensagem para o agente tradutor ..... 41
Figura 4.5 Protocolo para filtrar mensagens no agente tradutor ..... 43
Figura 4.6 Protocolo para ler uma mensagem no agente tradutor ..... 44
Figura 4.7 Protocolo para alinhamento de mensagem ..... 45

Figura 5.1 Interação entre agentes - Passo 1 ..... 56
Figura 5.2 Interação entre agentes - Passo 2 ..... 57
Figura 5.3 Interação entre agentes - Passo 3 ..... 58
Figura 5.4 Primeira tela do agente tradutor ..... 65
Figura 5.5 Segunda tela do agente tradutor ..... 65
Figura 5.6 Interface de login do usuário ..... 65
Figura 5.7 Agente usuário informando a impressora que executou a tarefa ..... 66
Figura 5.8 Primeira tela do servidor ..... 66
Figura 5.9 Segunda tela do servidor ..... 66

## LISTA DE TABELAS

Tabela 1.1 Etapas e tarefas do roteiro de pesquisa ..... 13

Tabela 2.1 Classificações de ontologias ..... 33

Tabela 4.1 Metadados de uma mensagem do agente tradutor ..... 42

Tabela 5.1 Impressoras do ambiente de teste ..... 48
Tabela 5.2 Computadores do ambiente de teste ..... 49
Tabela 5.3 Permissões do ambiente de teste ..... 49
Tabela 5.4 Ontologia GML v.1 ..... 50
Tabela 5.5 Padrão HP ..... 51
Tabela 5.6 Padrão Epson ..... 53
Tabela 5.7 Padrão Lexmark ..... 55
Tabela 5.8 Ontologia de permissão ..... 60
Tabela 5.9 Similaridade entre classes padrão HP e padrão Epson ..... 62
Tabela 5.10 Similaridade entre classes padrão Epson e padrão Lexmark ..... 62
Tabela 5.11 Similaridade entre classes padrão Lexmark e padrão HP ..... 63
Tabela 5.12 Similaridade entre propriedades de dados padrão HP e padrão Epson ..... 63
Tabela 5.13 Similaridade entre propriedades de dado padrão Epson e padrão Lexmark 64
Tabela 5.14 Similaridade entre propriedades de dado padrão Lexmark e padrão HP .... 67

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| IoT | Internet of Things |
| MQTT | Message Queue Telemetry Transport |
| CoAP | Constrained Application Protocol |
| IERC | European Research Cluster on the Internet of Things |
| FIPA | Foundation for Intelligent Physical Agents |
| OWL | Web Ontology Language |
| S-OWL | Semantic Markup for Web Services |
| RFID | Radio-Frequency Identification |
| IP | Internet Protocol |
| SOA | Service Oriented Architecture |
| M2M | Machine to machine |
| REST | Representational state transfer |
| XML | Extensible Markup language |
| BDI | Belief–Desire–Intention Architecture |
| ACL | Agent communication language |
| KQML | Knowledge Query and Manipulation Language |
| KSE | Knowledge Sharing Effort |
| SMA | Sistemas Multiagentes |
| W3C | World Wide Web Consortium |
| DL | Description logic |
| API | Application programing interface |
| QoS | Quality of service |
| QoI | Quality of information |
| URI | Uniform Resource Identifier |

| | |
|---|---|
| HP | Hewlett-Packard |
| INF | Instituto de informática |
| UFRGS | Universidade Federal do Rio Grande do Sul |
| GML | Geography Markup Language |
| RAM | Random access memory |
| MB | Mega bytes |
| SNMP | Simple Network Management Protocol |
| IPP | Internet Printing Protocol |
| MiB | Management Information Base |
| OID | 0bject Identifier |

**SUMÁRIO**

**1 INTRODUÇÃO ..... 11**
**1.1 Visão Geral ..... 11**
**1.2 Roteiro de Pesquisa ..... 12**
**1.3 Contribuições do trabalho ..... 13**
**1.4 Organização ..... 13**
**2 CONCEITOS BÁSICOS ..... 15**
**2.1 Internet das Coisas ..... 15**
**2.2 Agentes ..... 18**
2.2.1 Teoria dos agentes ..... 18
2.2.2 Arquitetura de agentes ..... 20
2.2.3 Linguagens de agente ..... 21
**2.3 Sistemas Multiagentes ..... 22**
2.3.1 Coordenação ..... 24
2.3.2 Cooperação ..... 25
2.3.3 Competição ..... 27
**2.4 Ontologias ..... 28**
2.4.1 OWL ..... 29
2.4.2 Reasoners ..... 30
2.4.3 Alinhamento de ontologias ..... 30
2.4.4 OWL-S ..... 32
**3 TRABALHOS RELACIONADOS ..... 34**
**4 IDEIA CENTRAL ..... 37**
**4.1 Conjuntos de ontologia ..... 37**
**4.2 Agente Tradutor ..... 39**
**4.3 Agente de Alinhamento ..... 43**
**5 IMPLEMENTAÇÃO ..... 47**
**5.1 Estudo de caso ..... 47**
5.1.1 Descrição do ambiente ..... 48
5.1.2 Descrição dos conjuntos de ontologia ..... 50
5.1.3 Interação entre os agentes ..... 56
5.1.4 Agente Impressora ..... 58
5.1.5 Agente Servidor ..... 59
5.1.6 Agente Usuário ..... 61
**5.2 Resultados ..... 62**
**6 CONCLUSÃO E TRABALHOS FUTUROS ..... 68**
**REFERÊNCIAS ..... 69**

# 1 INTRODUÇÃO

## 1.1 Visão Geral

Um assunto que é recorrente em computação hoje em dia é Internet das Coisas (IoT). Esse conceito despertou o interesse da comunidade científica, comercial e dos usuários em geral devido a sua enorme quantidade de aplicações e o modo como a tecnologia vai mudar a maneira que interagimos com o ambiente ao nosso redor.

Em (ATZORI; IERA; MORABITO, 2010), Atzori cita diversas aplicações em que o conceito de Internet das coisas pode ser usado, como por exemplo, no transporte e logística, provendo monitoramento em tempo real de entregas ou sistemas de navegação mais inteligentes, também existem aplicações no campo da saúde, como por exemplo, monitoramento de pacientes, coleta de dados médicos, gestão inteligente de recursos médicos (seringas, medicamentos,...). Podemos também encontrar aplicações em ambientes inteligentes, como casas inteligentes, fábricas inteligentes ou até mesmo museus inteligentes, que podem definir relações entre os objetos ou sugerir áreas do museu a partir do conhecimento dos interesses dos usuários. Devido a essa ampla quantidade de aplicações existentes, diversos dispositivos foram criados e com eles uma grande quantidade de protocolos e formatos de mensagens específicos, que visam suprir necessidades pontuais para esses dispositivos. Isso acabou gerando problemas de interoperabilidade em IoT, uma vez que, em um cenário de uma casa inteligente com sensores de luminosidade, qualquer adição ou substituição de um sensor de diferentes fornecedores, poderia gerar um problema de interoperabilidade entre protocolo ou formato de dados na aplicação.

Esse problema foi estudado em diversas pesquisas, como por exemplo (SUTARIA; GOVINDACHARI, 2013) que apresenta diversos protocolos que surgiram, buscando pela interoperabilidade entre protocolos especificos(no caso, MQTT e CoAP). Outro trabalho que tratou desse problema foi (VEER; WILES, 2008) que propõe uma padronização dos protocolos seguindo algumas regras. Além disso, em (SERRANO et al., 2015) é falado que essa camada somente será totalmente interoperável, quando houver uma conversão no uso desses protocolos pelos fornecedores de dispositivos IoT.

Apesar disso, notou-se com o tempo que mesmo zelando pela interoperabilidade entre os protocolos, formaram-se "ilhas" de dados nas diversas aplicações de Internet das coisas, por exemplo, uma residência que tenta se comunicar com o automóvel para saber quanto tempo determinado morador vai demorar pra chegar em casa, pode não conseguir

se comunicar pois suas representações do ambiente são diferentes (a menos que as aplicações sejam explicitamente desenvolvidas para isso), apesar de terem conceitos iguais, como a posição espacial do usuário. Nesse contexto, esse trabalho visa uma solução para esse tipo de problema de interoperabilidade, conhecido como interoperabilidade semântica.

Para isso, foi implementado um agente que realiza a tradução de um conceito de um domínio para um conceito em outro domínio, com auxilio de ontologias e algoritmos de alinhamento de ontologias. Agentes se encaixam muito bem nesse problema, pois é necessário autonomia e adaptabilidade que podem ser conferidos usando essa abordagem. O agente foi desenvolvido seguindo o padrão FIPA, provendo interoperabilidade a nível de agente. Para fornecer uma linguagem comum de acesso ao agente, foi utilizada a arquitetura blackboard, através de consultas SPARQL e um protocolo comum entre os agentes para acessar o quadro visa auxiliar o acesso das demandas por outros dispositivos.

Esse trabalho tem dois principais objetivos:

- *Objetivo 1:* Propor um agente que segue a arquitetura blackboard visando solucionar o problema de interoperabilidade semântica em IoT.
- *Objetivo 2:* Aplicar a solução encontrada e identificar pontos positivos e pontos que devem ser melhorados.

## 1.2 Roteiro de Pesquisa

O desenvolvimento do trabalho foi dividido em etapas, cada etapa apresenta um conjunto de tarefas que foram necessárias para concluí-la. A tabela 1.1 possui duas colunas. A primeira coluna representa a denominação da etapa na pesquisa e a segunda coluna apresenta a lista de tarefas realizadas na etapa que se refere.

A etapa de *Revisão Teórica* levantou os principais conceitos que nortearam o trabalho, e o estado da arte das pesquisas em interoperabilidade em Internet das Coisas. Nessa etapa foi definido qual seria o escopo do trabalho e quais contribuições eram visadas com ele. Na etapa de *Resolução do problema* surgiram diversas ideias para solucionar o problema, até chegar na solução do agente tradutor. Também foram desenvolvidos alguns módulos de ontologias que são necessários para representar um dispositivo em IoT.

A etapa de *Teste* foi realizada após a etapa de *Resolução do problema*. Nesse trabalho foi utilizado como estudo de caso um sistema de impressão, onde cada impressora

Tabela 1.1: Etapas e tarefas do roteiro de pesquisa

| *Etapa* | *Tarefas* |
|---|---|
| Revisão Teórica | - Fundamentos teóricos utilizados durante a pesquisa.<br>- Estado da arte das pesquisas em interoperabilidade em IoT.<br>- Definição do escopo do trabalho. |
| Resolução do Problema | - Definir módulos de ontologias necessárias para um agente em IoT.<br>- Concepção e Análise de um agente que possibilitasse a interoperabilidade semântica em IoT. |
| Teste | - Definição de um ambiente de teste.<br>- Definição das ferramentas utilizadas no desenvolvimento.<br>- Implementação dos agentes utilizados nos testes. |
| Análise | - Análise dos resultados obtidos do teste.<br>- Análise de melhorias e trabalhos futuros. |

Fonte: O Autor

tem um agente diferente, com ontologias diferentes, e o agente tradutor é utilizado para a comunicação entre impressoras, servidores e usuários. Por fim, a etapa de *Análise* os resultados dos testes foram analisados e justificados.

## 1.3 Contribuições do trabalho

Esse trabalho tem duas principais contribuições:

- Um agente tradutor para resolução do problema de interoperabilidade semântica em IoT.
- Um conjunto de ontologias que são utilizadas para demonstrar quais informações devem ser representadas em um ambiente IoT.

## 1.4 Organização

O conteúdo deste trabalho será apresentado em 6 capítulos. Esse primeiro capitulo apresentou uma introdução sobre o tema, quais os objetivos e de que forma está organizada a apresentação desse trabalho. No próximo capítulo serão apresentados os conceitos básicos de internet das coisas, agentes e ontologias, que são os conceitos bases desse trabalho.

O capitulo 3 compara alguns trabalhos que foram precursores a esse no estudo de interoperabilidade em IoT e como influenciaram na solução proposta neste projeto. No capitulo 4 o agente tradutor é apresentado, também são apresentados os módulos propostos de ontologia que um agente em internet das coisas deve considerar.

No quinto capítulo é proposta uma aplicação em um sistema de impressão para o agente tradutor traduzir as demandas de diferentes impressoras e servidores, é apresentado nesse capitulo o ambiente de teste, como foram moldados as ontologias para cada dispositivo e os resultados encontrados após a implementação. Por fim, o último capítulo apresenta a conclusão do trabalho, e como foram atingidos os objetivos específicos citados anteriormente.

# 2 CONCEITOS BÁSICOS

Nesse capítulo serão apresentados os conceitos que norteiam esse trabalho. A base da fundamentação teórica está dividida em quatro grandes conceitos: Internet das coisas, agentes, sistemas multiagentes e ontologias. Na primeira seção é apresentado o conceito de Internet das coisas, suas diferentes visões, as tecnologias que permitiram o seu surgimento e quais são os desafios que estão sendo resolvidos. Os conceitos relacionados à agentes são apresentados na segunda seção, onde são descritos os três pilares do estudo de agentes: a teoria dos agentes, arquiteturas de agentes e linguagens de agentes.

A terceira seção aborda os principais problemas de sistemas multiagentes, dentre eles, coordenação, cooperação e competição. A quarta seção descreve os conceitos relacionados a ontologia. Serão explicados a tecnologia OWL, o que é um *reasoner*, alguns algoritmos de alinhamento de ontologias. Além disso, mostrará o framework S-OWL que ajudou na implementação desse trabalho.

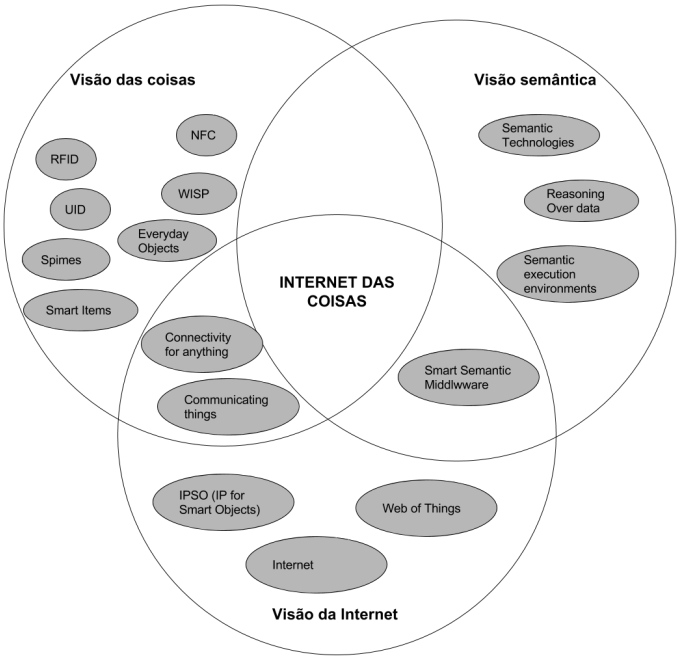
## 2.1 Internet das Coisas

Internet das coisas é um conceito que visa conectar todos os dispositivos entre si e também permitir a troca de informações desses dispositivos pela web, mudando a nossa interação com o ambiente. Hoje em dia, já podemos encontrar refrigeradores, carros, lâmpadas, entre outros dispositivos, que podem ter acesso a internet e usar seus dados para fornecer ao usuário facilidades.

Apesar de ser simples a definição, compreender o que é realmente IoT pode ser bastante difícil (ATZORI; IERA; MORABITO, 2010). Isso porque existem três perspectivas que devem ser analisadas. A figura 2.1 apresenta as principais tecnologias e conceitos que envolvem cada perspectiva. Na perspectiva das “Coisas” as tecnologias presentes visam identificar de forma única, acessar e manipular um objeto, um exemplo disso é o RFID, que foi o precursor do conceito de IoT. A perspectiva de “Internet” tem como foco fundamentar IoT sobre o protocolo IP, afim de tornar os dispositivos endereçáveis a qualquer outro. A visão “Semântica” tem como principais desafios a representação dos dispositivos e informações do sistema. A convergência dessas três perspectivas levam a internet das coisas.

A arquitetura que mais vem ganhando espaço em Internet das coisas é o SOA (Service Oriented Architecture). Em (KRAFZIG; BANKE; SLAMA, 2004) é proposta uma

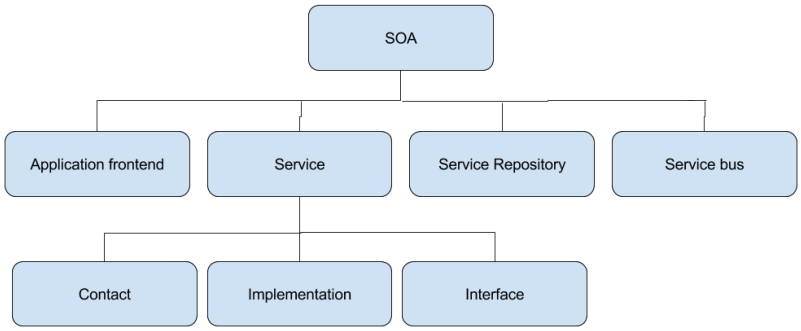
Figura 2.1: A convergência das três perspectivas resultam no conceito de Internet das Coisas




Fonte: Imagem adaptada de (ATZORI; IERA; MORABITO, 2010)

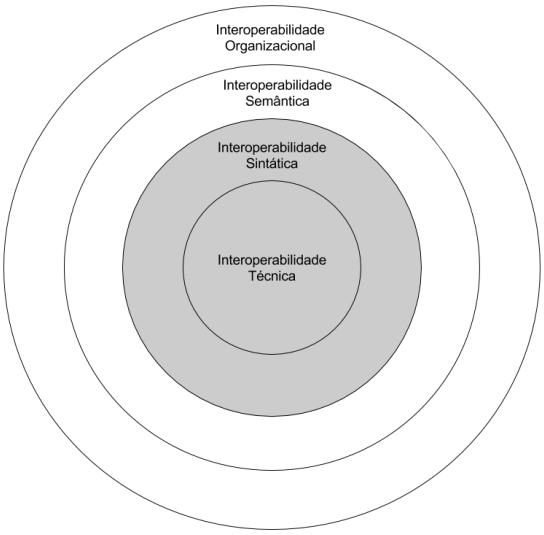
organização para arquitetura SOA que pode ser visualizada na figura 2.2. *Application frontend* é a forma que os usuários conseguem interagir com as outras aplicações SOA. *Service* representa serviços que são compostos por contrat (como o serviço é chamado), *implementation* (a implementação do serviço) e interface (interface de acesso ao serviço). O *service repository* é utilizado pelo usuário ou aplicação para obter maiores informações sobre o serviço que será utilizado. O *Service Bus* é responsável pela integração dos elementos da arquitetura SOA.

Figura 2.2: Organização da arquitetura SOA




Fonte: Imagem adaptada de (KRAFZIG; BANKE; SLAMA, 2004)

Figura 2.3: Classificação da interoperabilidade em IoT,segundo a IERC




Fonte: Imagem adaptada de (SERRANO et al., 2015)

Os dispositivos IoT são bastante heterogêneos e se utilizam da troca de mensagens para conseguir transmitir informações, mas por ser um conceito ainda bastante recente, cada fornecedor possui seus próprios padrões, e a convergência desses padrões pode demorar para acontecer. Além disso, a grande quantidade de domínios e de dispositivos que o conceito de IoT pode ser aplicado exige padrões diferentes, que atendam as necessidades específicas de cada dispositivo.

Nesse contexto, o estudo da interoperabilidade em IoT exerce um papel muito importante, é necessário procurar soluções que permitam a comunicação entre sistemas e entre dispositivos. A IERC (European Research Cluster on the Internet of Things), um orgão europeu com objetivo de explorar o grande potencial de IoT na Europa, classificou a interoperabilidade em quatro camadas (SERRANO et al., 2015). Para alcançar a camada mais externa é necessário ter alcançado todas as outras internas.

Assim como mostra a figura 2.3, a camada mais interna é a interoperabilidade técnica. Essa interoperabilidade tem como objetivo a comunicação M2M (machine to machine). Seu foco principal é prover a interoperabilidade entre os protocolos. Atualmente, diversos trabalhos já visam solucionar o problema proposto nessa camada, como citado anteriormente, a maioria deles se baseia em padronizações e extensões de protocolos para se adaptar a outros. A camada acima é a interoperabilidade sintática que trabalha para prover interoperabilidade entre os formatos dos dados, pois os protocolos transmitem informações que devem estar em um formato de forma que estejam acessíveis a quem foi transmitido, para isso é necessário o estudo de formatos interoperáveis.

No estado da arte dessa camada foram desenvolvidos diversos sistemas que proviam aos dispositivos informações no formato XML. É possível citar os trabalhos (COLLINA; CORAZZA; VANELLI-CORALLI, 2012) que é um web server baseado no protocolo MQTT e (CASTELLANI et al., 2011) que é um webserver baseado no protocolo CoAP.

Na interoperabilidade semântica, não basta apenas o dado estar em um formato interoperável, mas também deve ter um significado para ambos os dispositivos. Um exemplo seria de uma aplicação que utiliza temperaturas de diversos lugares para fornecer algum serviço ao usuário, a temperatura pode vir de um termômetro na rua, de um climatizador de um hospital ou até de um sensor de um carro, mas a aplicação precisa saber que aquela informação é uma temperatura. Segundo a IERC (SERRANO et al., 2015), essa camada é o próximo passo nas pesquisas de interoperabilidade em IoT e se faz necessário o uso de tecnologias semânticas, especialmente de web semântica, para resolver esses problemas. Por fim, a interoperabilidade organizacional é a capacidade organizações trocarem informações, independente das diferenças geográficas e culturas.

Este trabalho explora a interoperabilidade semântica em IoT. Para isso, foi necessário definir módulos de ontologias que representassem os ambientes semanticamente em IoT e um agente que pudesse lidar com as diferentes representações de conhecimento dos dispositivos, com objetivo de permitir interação entre dispositivos com conhecimentos diferentes.

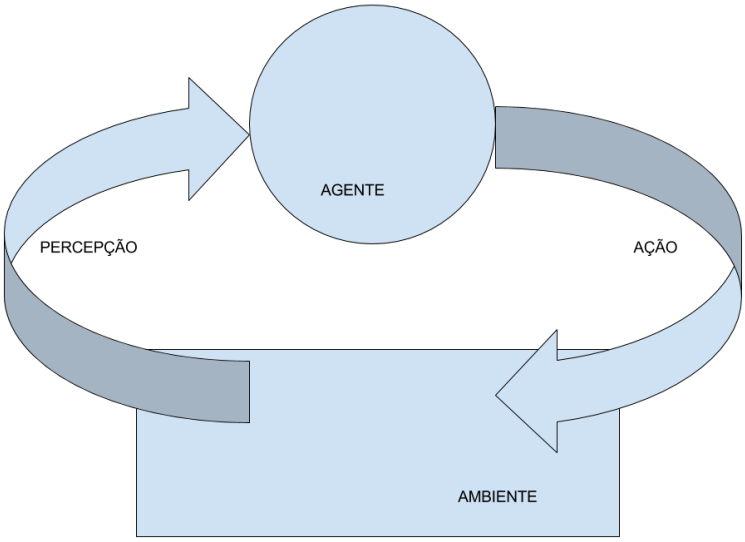
## 2.2 Agentes

O estudo de agentes é baseado basicamente em três categorias: teoria dos agentes, arquitetura de agentes e linguagens de agente. A teoria dos agentes busca compreender o que é um agente, suas principais propriedades e como formalizá-los. Arquiteturas de agentes visam projetar sistemas de hardware ou software que satisfazem as propriedades dos agentes definidas na teoria dos agentes. Por fim, as linguagens de agente permitem a implementação dos agentes, seguindo suas conceitualizações.

### 2.2.1 Teoria dos agentes

Existem muitas definições de agentes na literatura. Este trabalho utiliza a definição de Russel e Norvig (2003), “um agente é tudo que pode ser considerado capaz de perceber

Figura 2.4: Percepção e ação no agente




Fonte: O Autor

seu ambiente por meio de sensores e de agir sobre esse ambiente por meio de atuadores", podemos observar a ilustração desse conceito na figura 2.4. Podemos notar pela figura 2.4 que o termo percepção é usado para se referir às informações recebidas do ambiente pelos sensores e que o termo ação se refere à intervenção de um atuador no ambiente.

Podemos definir mais formalmente o conceito de agente como uma função: função de agente. Basicamente, essa função mapeia qualquer sequência de percepções para uma ação. Se a função fosse representada como uma tabela, possivelmente ela seria muito grande, ou até infinita, isso tornaria impossível uma implementação concreta. Para resolver esse problema se desenvolvem arquiteturas de agentes, que serão abordadas na próxima subseção.

Nesse trabalho estamos interessados em um tipo especial de agente, o agente racional. Ainda segundo Russel e Norvig, um agente racional é aquele que faz tudo certo, ou seja, um agente que obtém sucesso nas ações que dependem somente dele. Para medir esse sucesso é definido o conceito de medida de desempenho, que normalmente é projetada usando o resultado desejado no ambiente. Todo ambiente tem um estado, e na medida que o agente age sob o ambiente esse estado muda. Se o estado que o ambiente adquiriu depois das ações do agente estiverem dentro do esperado, a medida de desempenho será alta e o agente será considerado racional.

Deve-se notar que obter sucesso em suas ações não quer dizer ser perfeito. As decisões de um agente racional dependem apenas das percepções obtidas até o momento da

ação, a perfeição está mais ligada à onisciência, onde o agente deveria ter conhecimento de todo o ambiente e até prever o futuro, assim seria impossível atender essas especificações.

Além de racionalidade, outra característica de um agente é a autonomia. Um agente ser autônomo significa que seu conhecimento é baseado basicamente de suas percepções. Um agente completamente autônomo na prática não é aplicável, pois isso implica o agente realizar ações ao acaso até aprender a agir de forma racional.

### 2.2.2 Arquitetura de agentes

Um agente pode ser classificado como cognitivo ou reativo. Agentes cognitivos são os mais próximos de agentes racionais o quanto possível, ou seja escolhem a melhor ação dentre as possíveis. Para isso, devem ter uma representação explicita de conhecimento do mundo. Agentes chamados reativos são os mais simples. Habitualmente baseados no padrão de comportamento estímulo-resposta. Segundo (OLIVEIRA, 1996), os agentes tem um grau de racionalidade, que classifica seu comportamento entre os dois tipos de agente.

- **Arquiteturas cognitivas ou deliberativas**: Nesse tipo de arquitetura os agentes apresentam alguma representação simbólica do mundo, entende-se que suas decisões (deliberações) e se baseiam em raciocínio logico sobre essa representação. Além da representação simbólica do mundo, um agente deliberativo possui um plano e uma função de utilidade para a ação. Existem duas classificações possíveis para uma arquitetura cognitiva: Arquiteturas funcionais e arquiteturas baseadas em estados mentais.
    - *Arquiteturas funcionais*: Os módulos de cada agente representam as funções que um agente pode fazer. Duas arquiteturas desse tipo são definidas em (DEMAZEAU, 1995) e (STEINER, 1996).
    - *Arquiteturas baseadas em estados mentais*: Nessa arquitetura, alguns componentes que podem definir o estado mental do agente: crenças, capacidades, escolhas e compromissos. A principal arquitetura que representa essa classe é a BDI, que considera como estado mental a crença, desejo e intenção de um agente. As crenças (*beliefs*) representam o ambiente que é percebido pelo agente, os desejos (*desires*) representam o estado emocional do agente

e os seus objetivos, por fim, as intenções (*intentions*) representam o plano do agente para chegar em seus objetivos. Nessa arquitetura existem duas principais ações do agente. A primeira é decidir sobre quais objetivos queremos agir e a segunda é como agir sobre eles.

- **Arquiteturas reativas**: Também são chamadas de arquiteturas não-deliberativas, são arquiteturas que tomam decisões baseando-se apenas em respostas do ambiente, onde normalmente o controle é modelado como regras "evento-ação", e o agente pode ser representado como um autômato finito simples. Por não precisarem planejar ações, possuem pouca representação, ou nenhuma representação do ambiente.
- **Arquiteturas híbridas**: Arquiteturas deliberativas apresentam grandes dificuldades ao serem implementadas em sistemas que exigem uma resposta em tempo ágil, pois todas as ações do agente dependem de suas crenças e não apenas de suas percepções. Por outro lado, arquiteturas reativas apresentam a limitação de não planejarem ações futuras. Visando resolver os problemas das duas arquiteturas, uma arquitetura hibrida foi proposta. Nessas arquiteturas devem estar presentes características tanto reativas quanto deliberativas. Um exemplo de um agente com arquitetura hibrida poderia ser um que segue um planejamento, entretanto após a percepção de algo no ambiente, rapidamente interrompe o que estava fazendo para responder com uma ação, voltando depois ao seu planejamento.

Existe uma competição chamada Robocup, onde competidores do mundo todo competem fazendo um time de robôs de futebol. Para controlar esses robôs, são utilizados agentes. No artigo (ZHANG et al., 2000) são analisadas algumas arquiteturas que foram utilizadas durante a competição do ano de 2000 e o seu desempenho quando disputavam com outras arquiteturas. A conclusão que esse trabalho chegou foi que, as arquiteturas que tiveram melhor classificação, são consideradas hibridas, pois atribuíam tarefas de correr em direção à bola, se mover ao redor do campo e chutar como características reativas, e definiam estratégias globais como característica deliberativa. As arquiteturas somente reativas não jogavam com muita estratégia, e os outros agentes percebiam rapidamente o padrão de jogo. Já as arquiteturas deliberativas não obtiveram um bom desempenho, pois uma partida de futebol tem uma natureza muito dinâmica, e os agentes deliberativos demoravam muito para tomar algumas decisões.

### 2.2.3 Linguagens de agente

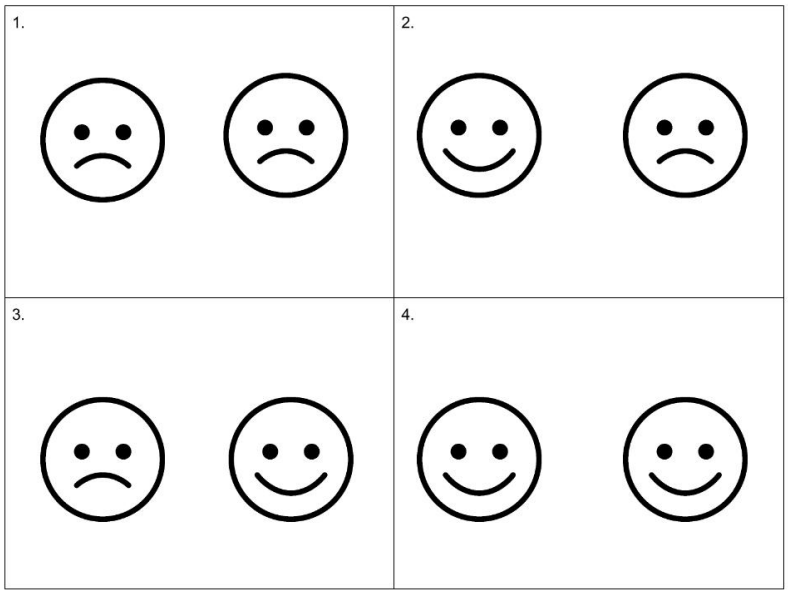
Os agentes são dotados de habilidade social, ou seja, tem a capacidade de se comunicar com outros agentes. Essa interação é realizada por meio de uma linguagem de comunicação entre agentes. Na literatura, essas linguagens são conhecidas como ACL (Agent Communication Languages). É baseada na teoria dos atos comunicativos, que é uma teoria proposta por Austin em 1962 (AUSTIN, 1975), que a fala também representa uma ação(ou ato comunicativo), ou seja, quem fala não apenas declara sentenças, mas também pode fazer pedidos, promessas e dar ordens. No caso de agentes, quando um agente manda uma mensagem para outro agente, esse envio da mensagem também representa um ato comunicativo. O ato comunicativo em agentes é importante pois demonstra a intenção do agente locutor e tem impacto no modo que a informação será tratada pelo agente receptor. Atualmente, para agentes existem duas linguagens de interação principais: KQML e FIPA ACL.

- **KQML**: Foi desenvolvida pela KSE e é um acrônimo para Knowledge Sharing Effort. Define um conjunto de atos comunicativos para expressar os desejos do agente locutor. Toda mensagem deve informar um campo "performativa"que será substituido por algum dos atos comunicativos da KQLM, e alguns parâmetros, como por exemplo quem enviou, quem irá receber ou o conteúdo da mensagem. O grande problema da KQML é que existem atos comunicativos cujo os nomes não correspondem diretamente as suas reais funções. Além disso, existem termos ambíguos e faltam atos comunicativos que notaram-se necessários para diversos agentes.
- **FIPA ACL**: A FIPA ACL foi criada pela FIPA (Foundation for Intelligent Physical Agents), que é uma fundação sem fins lucrativos localizada na Suíça que cria especificações para promover interoperabilidade entre agentes. Diferentemente da KQML, a FIPA ACL tem grande preocupação com a semântica dos atos comunicativos. A sintaxe é muito parecida com KQML, mas os atos comunicativos são diferentes.

## 2.3 Sistemas Multiagentes

Sistemas multiagentes (SMA) surgiram do desenvolvimento de inteligência artificial distribuída (WEISS, 1999), e envolvem a solução de problemas de forma distribuída,

Figura 2.5: Cooperação e competição entre agentes




Fonte: O Autor

utilizando diversos agentes que interagem entre si. Agentes podem querer se comunicar para alcançar melhores objetivos para si ou para o sistema. Um sistema multiagente pode ser um sistema heterogêneo, ou seja, podem existir mais de um tipo de agente, que tem funções, observações ou ações no ambiente diferentes um dos outros. Além disso, deve ser capaz de oferecer mecanismos e métodos que permitam coordenação desses diversos agentes no sistema para cumprir seu objetivo.

O comportamento entre agentes em um sistema pode ser competitivo, onde os agentes competem entre eles para resolver o problema, ou pode ser cooperativo, onde os agentes cooperam para resolver juntos o problema que estão propostos a solucionar. Para ambos comportamentos, o sistema precisa de coordenação, que é alcançada por meio de protocolos que serão apresentados neste capítulo. A figura 2.5 mostra um exemplo de um sistema multiagente, mostrando as relações de cooperação e competição entre os agentes, onde os quadros 1 (quando um agente tem resultado ruim, os dois são prejudicados) e 4 (quando um agente tem resultado bom os dois são beneficiados) demonstram casos de cooperação entre os agentes e os quadros 2 e 3 (o sucesso de um agente depende do fracasso do outro) apresentam competição entre os agentes.

Assim como existem agentes reativos e cognitivos, também existem sistemas multiagentes reativos e cognitivos. Um sistema multiagente reativo é composto por agentes reativos, e são baseados em algoritmos de inteligencia coletiva. Apesar de individualmente realizarem tarefas simples e apenas baseadas em “evento - ação” e não apresen-

tarem nenhuma representação do ambiente, coletivamente apresentam comportamento inteligente, todo o seu conhecimento esta implícito em suas regras de comportamento. Entretanto para que esse tipo de SMA tenha resultados bons, é necessário um grande número de agentes e consequentemente um grande número de mensagens trocadas.

Os sistemas multiagentes cognitivos são compostos por agentes cognitivos. Individualmente são mais sofisticados que os agentes reativos, coletivamente se utilizam de relações sociais e não precisam de muitos agentes para apresentar um comportamento inteligente.

### 2.3.1 Coordenação

Coordenação é a propriedade de um sistema de agentes executar uma atividade em um sistema compartilhado (WEISS, 1999). Em um ambiente recursos são limitados e compartilhados, assim agentes precisam coordenar suas tarefas a fim do sistema ser coerente. O problema da coordenação de agentes poderia ser resolvido centralizando o controle e os dados do sistema, mas isso limitaria uma das características principais dos agentes, a autonomia. Para evitar isso, é necessário que seja informado explicitamente as dependências entre as tarefas a serem coordenadas, entretanto em sistemas multiagentes, normalmente os agentes tem apenas uma visão parcial do ambiente.

Desta forma, se um agente deseja obter uma informação diferente de sua região é possível fazê-lo de forma indireta por meio de uma observação das mudanças ocorridas no ambiente que o agente pertence. Além disso, é possível obter a informação de forma direta, interagindo com outro agente que pertence ao ambiente que contém a informação desejada. Para essa última opção é necessário que ambos os agentes compartilhem uma linguagem em comum e ambos devem ter conhecimento da existência do outro.

É possível considerar duas situações na coordenação de forma direta. Tomemos como exemplo dois agentes, o agente A e o agente B, que foram desenvolvidos para resolver o problema de limpar uma casa. Na primeira situação é possível fazer os agentes com funcionalidades diferentes, por exemplo, um sabe limpar o chão e o outro sabe limpar os móveis, fazendo cada um responsável por uma parte da casa. Um bom desempenho do agente A resultará em um bom desempenho do agente B, assim como o bom desempenho do agente B resultará em um bom desempenho do agente A. Esse tipo de coordenação é chamada de cooperação. Podemos ter outra situação, onde ambos os agentes sabem limpar o chão e os móveis, competindo entre si para saber quem limpa mais áreas da casa,

nesse caso um bom desempenho do agente A resulta em um mau desempenho do agente B, e consequentemente o bom desempenho do agente B resulta em um mau desempenho do agente A. Esse tipo de relação de coordenação é chamada de competição.

### 2.3.2 Cooperação

Como foi descrito na subseção anterior, a cooperação é uma forma de coordenação, onde o bom funcionamento de um agente reflete no bom funcionamento do outro agente. Pode-se também dizer que a cooperação é a coordenação de agentes não antagônicos (WEISS, 1999). Nesse tipo de coordenação os agentes resolvem os problemas como um time, onde cada agente pode resolver parte do problema. Para isso, os problemas devem ser decompostos sucessivamente em subproblemas até se tornarem simples o suficiente para um agente resolver. Depois da resolução dos subproblemas os agentes fazem a síntese de suas soluções, obtendo a solução final. A decomposição em subproblemas acaba criando relações de dependência entre as tarefas. Uma solução encontrada para isso são os protocolos de cooperação. Nesse trabalho vamos citar dois desses protocolos: Rede de contato e Blackboard.

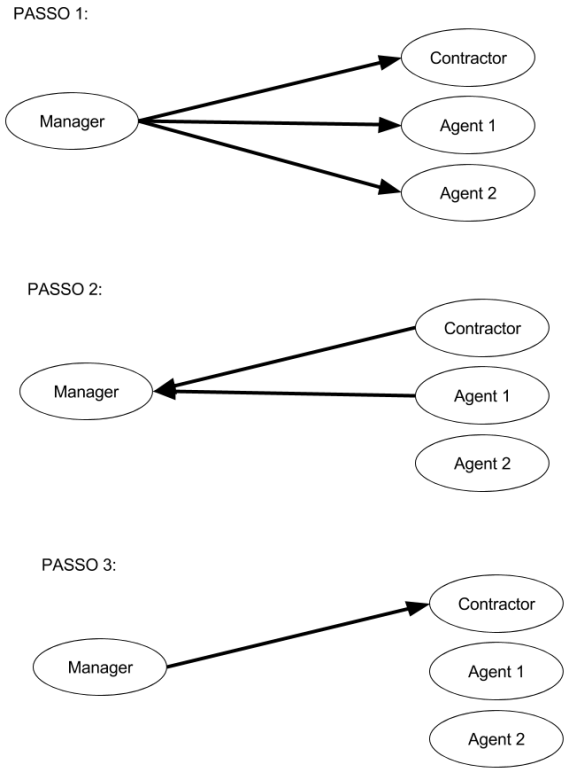
O protocolo “Rede de Contrato” foi proposto por Reid G. Smith em (SMITH, 2006), em 1980. Nesse protocolo cada agente assume um dos dois papéis diferentes, ou *manager* ou *contractor*. O agente que “quebrou” o problema em subproblemas é o *manager*, é responsável por monitorar a execução das tarefas e processar os resultados de sua execução. Os agentes que se comprometeram com o *manager* em resolver esses problemas são os *contractors*, que são responsáveis pela execução da tarefa.

A principio, um agente pode assumir ambos os papéis em um único sistema, podendo ser *manager* de um agente e *contractor* do mesmo, dependendo do contrato, por essa razão na implementação devem ser considerados ambos os papéis.

A figura 2.6 apresenta um exemplo de como funciona o protocolo: No primeiro passo, o *manager* anuncia alguma tarefa pendente aos três agentes da direita na imagem. Ao receberem o anuncio, cada agente verifica a possibilidade de realizar a tarefa proposta. No segundo passo, caso o agente tenha a possibilidade de realizar a tarefa, ele retorna ao *manager* oferecendo o seu serviço, caso contrario, o anuncio é ignorado. O *manager*, recebendo as ofertas de serviço, escolhe a que melhor atende suas expectativas e delega a tarefa para um agente, como mostrado no passo três.

O outro protocolo, o blackboard ("Quadro Negro"), foi proposto em 1985 por

Figura 2.6: Protocolo Rede de Contrato




Fonte: O Autor

Barbara Hayes-Roth (HAYES-ROTH, 1985). Inicialmente foi proposto como uma arquitetura, a ideia era usar a metáfora do quadro negro sendo utilizado por um grupo de especialistas que trabalham cooperativamente para realizar uma tarefa. Toda a vez que um especialista precisa que seja realizada uma tarefa que ele nao pode realizar, escreve no quadro negro, para que todos os outros especialistas que estão olhando para o quadro negro possam ajudá-lo. O quadro negro nesse caso funciona como uma área compartilhada onde os agentes podem trocar informações. Os agentes são fontes de conhecimento e observam esse quadro, até o momento em que podem contribuir para o sistema com seu conhecimento. Um agente pode escrever e ler do quadro, filtrar informações e se registrar no quadro.

É importante destacar dois pontos importantes desse protocolo:

- Um agente não pode acessar uma determinada estrutura de dados, se outro agente já a bloqueou.
- Nessa arquitetura os agentes são capazes de interagir com agentes não previstos inicialmente pelo arquiteto do sistema, basta compartilharem uma mesma linguagem de interação comum.

Ambos os protocolos Rede de contato e blackboard são ideias bem simples, por isso são muito estudados e têm diversas implementações na literatura. A arquitetura blackboard é muito utilizada em robótica para coordenar as ações de diversos robôs, em (XU; BRUSSEL, 1997) o autor desenvolve uma arquitetura blackboard para os agentes de seus robôs móveis se comunicarem. Em (RITCHIE, 1990) um blackboard é utilizado para gerenciamento do trânsito, fazendo as estradas se comunicarem entre si e reportarem incidentes ocorridos para fazer o trânsito se adaptar. Foram encontradas diversas aplicações do protocolo rede de contrato para aplicações de gerenciamento de produções, o trabalho (BAKER, 1988) reporta o uso do protocolo para controle de manufatura.

### 2.3.3 Competição

Na competição um agente trabalha individualmente para atingir seus objetivos, sabendo que outros agentes diminuirão suas medidas de desempenho com o seu sucesso. Com essas definições, para haver competição, antes de tudo, um agente deve ter conhecimento da presença dos outros agentes que ele esta competindo, dessa forma pode prever os seus comportamentos e tomar uma decisão que o favoreça.

Para alguns autores, a coordenação de agentes concorrentes decorre da negociação. Nesse caso, quando um agente disputa um recurso com outros, na negociação, não importa para o agente se a utilização deste recurso vai privar os outros de usá-lo e consequentemente prejudicar o sistema em geral, a única preocupação desse agente é o seu desempenho, são agentes de auto-interesse.

## 2.4 Ontologias

Desde a Grécia antiga Platão, Aristóteles e Parmenides, se dedicavam ao estudo da propriedades gerais das coisas, procuravam saber o que é essencial e o que não é. Mesmo no campo das ideias era difícil definir uma ontologia.

Ontologias ganharam força na computação, a partir do advento do conceito de Web Semântica, que dado o grande crescimento da quantidade de dados da internet, se faz necessário acrescentar semântica a esses dados para que a informação não seja apenas compreendida por humanos, mas também pelas máquinas, desta maneira os computadores podem auxiliar em tarefas que hoje fazemos manualmente. Neste trabalho o conceito utilizado de ontologia é definido como uma especificação formal e explícita de uma conceitualização compartilhada. Esse conceito de ontologia, admite que deve haver uma visão compartilhada entre os diversos interessados. No subcapítulo sobre Internet das Coisas, a camada de interoperabilidade semântica envolve o uso de ontologias para sua solução. É necessário que uma ontologia seja formal, ou seja, em uma linguagem e formato compreeensiveis pela máquina. Na próxima subsessão será apresentada a linguagem OWL, que é uma linguagem recomendada pela W3C para representar ontologias.

Existem diversas aplicações onde ontologias podem ser usadas para solucionar problemas. Em (MORAIS; AMBRÓSIO, 2007), os autores citam diversas razões para se utilizar ontologias, em comunicação são usadas para determinar consenso entre os diversos participantes, especialmente sobre termos técnicos ou regras, também formalização as ontologias são utilizadas para eliminar ambiguidades que possam existir, e por fim podem ser usadas para reutilização, pois uma vez que um conhecimento de domínio publico foi abstraído, pode ser reaproveitado por outros usuários.

Muitas ontologias são propostas na mesma área de conhecimento, mas com finalidades diferentes. As ontologias podem ser classificadas de acordo com diversas características, as classificações das ontologias e descrição são mostradas na tabela 2.1

O próximo subcapítulo apresenta a linguagem OWL, que é a linguagem mais uti-

lizada para representar ontologias e é uma recomendação da W3C. Após apresentação do OWL é introduzido o conceito de reasoner, que é um importante componente da web semântica para poder se aproveitar de todo conteúdo das ontologias. Por fim, são apresentados algoritmos de alinhamento de ontologia, que é a técnica utilizada neste trabalho para obter interoperabilidade semântica.

### 2.4.1 OWL

OWL (Web Ontology Language) é uma linguagem para definir e instanciar ontologias web baseada em RDF. Ela é composta por classes, propriedades e indivíduos. Uma ontologia descreve um domínio de interesse, esse domínio pode ser grande, pequeno, pode ser muito abstrato ou concreto. Nesse domínio é possível identificar conceitos, indivíduos, propriedades desses individuos e relações entre eles. Em OWL são utilizados os conceitos de classes, propriedades e indivíduos para representar as entidades do domínio de interesse. As classes representam um conceito na ontologia, ou seja, é um conjunto que pode conter indivíduos, esses indivíduos são instâncias das classes. Os indivíduos podem ter propriedades, que relacionam eles a outros indivíduos (propriedades de objeto) ou a valores de dados (propriedades de dados). As propriedades podem ser funcionais, funcionais inversas, simétricas ou transitivas. Uma propriedade funcional implica que um individuo pode apenas se relacionar com um outro através daquela propriedade, dessa forma é possível identificar se dois indivíduos são o mesmo individuo, por meio de suas relações. A propriedade funcional inversa resulta que a sua propriedade inversa é funcional. Uma propriedade é simétrica quando a relação é equivalente para ambas as partes, por exemplo uma propriedade chamada “é irmão” que relaciona dois indivíduos (A e B), podemos dizer que A “é irmão” de B, assim como B “é irmão” de A. Por fim, uma propriedade pode ser transitiva, ou seja se existir uma propriedade “é maior” que é transitiva, e os indivíduos A, B e C se relacionando da seguinte forma: A “é maior” B, B “é maior” C. Com a propriedade transitiva, podemos inferir que A “é maior” C.

Existem três sublinguagens em OWL, que foram definidas para usuários que tem diferentes necessidades de expressividade. Quanto maior a expressividade da linguagem, a performance nos processos de inferência tendem a ser menores, então é necessário que quem for utilizar a linguagem OWL saiba bem qual a sua necessidade de expressividade:

- **OWL Lite**: é a representação mais simples da linguagem OWL, pode ser usada para

classificações hierárquicas ou restrições simples. É possível realizar, por exemplo, restrição de cardinalidade binária ou intersecção de classes.

- **OWL DL**: esse subconjunto da linguagem apresenta a maior expressividade e computável em um tempo finito. Existem algumas restrições para sublinguagem, por exemplo, uma classe não pode ser um indivíduo ou uma propriedade.
- **OWL Full**: é possível ter a máxima liberdade sintática e expressividade semântica com a linguagem, mas não é garantido que seja computável. Não é provável que softwares de inferência suportem a totalidade dessa linguagem.

Pode-se notar que cada sublinguagem é um extensão da anterior, então toda a ontologia OWL Lite válida, também é uma OWL DL válida e uma OWL Full válida, ou ainda toda OWL DL válida é também uma OWL Full válida. Entretanto o oposto disso não é verdadeiro, nem toda OWL DL válida é uma OWL Lite válida (note que é possível, mas não são todos os casos).

### 2.4.2 Reasoners

Nem sempre as informações nas ontologias estão explícitas, muitas vezes é necessário mecanismos de inferência como auxílio, esses mecanismos são encontrados nos reasoners. Um reasoner é um software capaz de inferir consequências lógicas em um conjunto de axiomas. Em (DENTLER et al., 2011) são citados três atributos que devem ser priorizados em um reasoner. O primeiro é o raciocínio, sendo o principal atributo que confere a funcionalidade básica de um reasoner. Usabilidade prática é o segundo atributo e se refere ao quanto foi implementada a API da OWL, além da disponibilidade e da licença do reasoner. Por fim, as métricas que são utilizados para mensurar a performance dos reasoners.

Um reasoner pode usar duas estratégias diferentes para inferir sobre os axiomas. A primeira estratégia é encadeamento para frente, onde se parte dos axiomas conhecidos, e a partir desses se derivam outros axiomas. Os possíveis objetivos para usar esse tipo de inferência seria computar um fechamento de axiomas, responder uma query particular ou classificação de ontologia. Outra estratégia é o encadeamento para trás, onde o reasoner parte de um fato e verifica a validade dele em determinada ontologia.

Em (ABBURU, 2012) é apresentada uma comparação entre diversos reasoners disponíveis tanto comercialmente como de forma livre. Nesse trabalho foi utilizado o

Hermit, que é um reasoner para ontologias OWL. O raciocínio é garantido utilizando um algoritmo de prova automática de teoremas, no caso o Hyper tableaux (BAUMGARTNER; FURBACH; NIEMELÄ, 1996), implementado usando a linguagem Java. É um reasoner Open source e bastante completo.

### 2.4.3 Alinhamento de ontologias

Como descrito anteriormente, uma ontologia é uma especificação formal e explícita de uma conceitualização compartilhada, entretanto não necessariamente única. Diferentes fornecedores podem apresentar diferentes ontologias (com vocabulários diferentes, estruturas diferentes, relações diferentes, entre outros) para representar os mesmos conceitos. Essa diversidade de ontologias causa problemas de interoperabilidade, uma vez que aplicações que usam ontologias diferentes, mas com os mesmos conceitos, não podem se comunicar.

Esse problema é o foco do alinhamento de ontologias, segundo (EUZENAT, 2009) alinhamento de ontologias é achar correspondências entre as entidades dessas ontologias, por meio do processo de casamento de ontologias (ontology matching). Esse processo tem como entrada duas ontologias (O1 e O2) e o objetivo é encontrar o alinhamento (A'). Pode ser fornecido também uma ontologia de alinhamento (A), juntamente com parâmetros e outros recursos externos.

Formalmente, o alinhamento de ontologias pode ser definido através de uma função align, considerando três parâmetros de entrada: duas ontologias O1 e O2, que serão as ontologias que terão suas entidades alinhadas, e uma entidade e1 que é um elemento pertencente ao conjunto de todas as entidades das ontologias. Essa função resulta em uma entidade e2, que representa a entidade alinhada com e1.

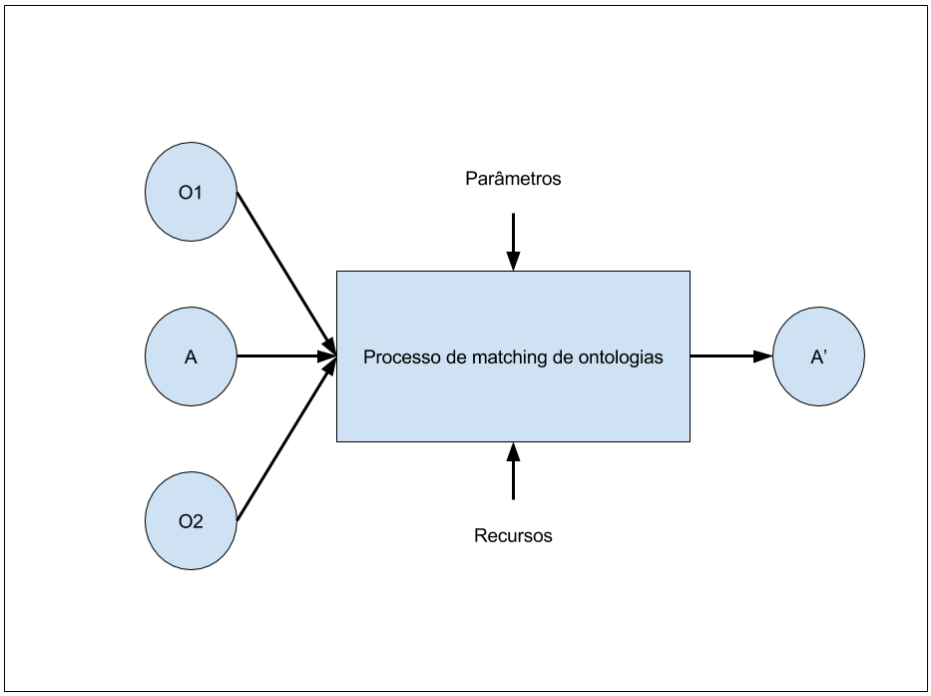
*align:* $E \times O \times O \rightarrow E$

*onde E é o conjunto de todas as entidades das ontologias (pode ser classes, propriedades de dados ou objetos e indivíduos) e O é o conjunto das ontologias.*

*align(e1,O1,O2) = e2*

O alinhamento de ontologias pode ser feito de maneira automática, semiautomática ou manual. Um alinhamento de ontologias automático é feito sem nenhuma intervenção do usuário em nenhuma parte do processo. No alinhamento semiautomático, o usuário intervém em algumas partes do processo, é muito utilizado, especialmente em aplicações que exigem precisão e sem perder muito desempenho. O alinhamento manual

Figura 2.7: Processo de matching de ontologias




Fonte: Imagem adaptada de (SILVA; GLUZ, )

é realizado pelo usuário, e exige um grande esforço dele para realizar, o uso de ferramentas gráficas ajuda a amenizar esse esforço.

Essa função align pode ser obtida por meio de uma função que retorne um grau de similaridade entre as entidades. Existem diversas técnicas para implementar essa função, pode ser por meio de termos, onde são utilizadas técnicas de mineração de dados que identificam termos em comum nas entidades, pode ser considerada também a estrutura da ontologia, onde são verificadas as relações e atributos das entidades, considera-se também a extensão da ontologia, ou seja, os indivíduos. Por fim, pode ser utilizada interpretação semântica para encontrar os alinhamentos (SILVA; GLUZ, ).

### 2.4.4 OWL-S

Nesse trabalho, os serviços em IoT são representados por meio de ontologias, como veremos posteriormente. Essas ontologias dos serviços estão aos moldes de OWL-S (Web Ontology Language for Services), que é um framework para a integração de web services e web semântica (OWL-S. . . , ). Basicamente é composto por três blocos: Service Profile, Service Model e Service Grounding. O service profile descreve o serviço mostrando os tipos de entrada, os tipos de saída, pré-condições e pós-condições do serviço.

No service model o serviço pode ser representado como atômico (caso seja invocado e executado em um único passo) ou composto (caso seja composto por várias etapas), dessa forma é possível indicar como o serviço funciona, qual informação enviar ou receber em cada etapa. O service grounding apresenta os protocolos de comunicação para acessar o serviço.

Tabela 2.1: Classificações de ontologias

| **Característica** | **Classificação** | **Descrição** |
|---|---|---|
| Grau de Formalismo | Altamente Informal | Expressas utilizando linguagem natural. |
| | Semi-informal | Expressas utilizando linguagem natural, mas com algumas restrições e estrutura. |
| | Semi-formal | Expressas utilizando linguagem artificial, definidas com certa formalidade. |
| | Rigorosamente formal | Expressas com semântica formal, teoremas e provas. |
| Aplicação | Autoria Neutra | Utilizadas para trazer portabilidade de informações de uma aplicação para outra. |
| | Especificação | São usadas para manutenção e documentação de aplicações |
| | Acesso comum à informação | Apresentam conhecimento compartilhado dos termos do vocabulário |
| Conteúdo | Terminológicas | Apresentam os termos que modelam um domínio de conhecimento especifico. |
| | Informação | Apresentam a estrutura do domínio de conhecimento. |
| | Modelagem de conhecimento | Definem os conceitos do conhecimento. |
| | Aplicação | Modelam o conhecimento de uma aplicação. |
| | Domínio | Modelam conceitos de um domínio especifico do conhecimento. |
| | Genéricas | Definem conceitos genéricos a vários domínios de conhecimento. |
| | Representação | Explicam conceitualizações que apoiam os formalismos de representação do conhecimento. |
| Função | Genéricas | Definem conceitos genéricos a vários domínios de conhecimento. |
| | Domínio | Modelam conceitos de um domínio especifico do conhecimento. |
| | Tarefas | Descrevem tarefas que são usadas especialmente no auxilio de resolução de problemas. |
| | Aplicação | Utilizam das ontologias de domino e tarefa para modelar uma aplicação que define regras sobre indivíduos quando executam determinada tarefa. |
| | Representação | Explicam conceitualizações que apoiam os formalismos de representação do conhecimento. |

Fonte: O Autor

# 3 TRABALHOS RELACIONADOS

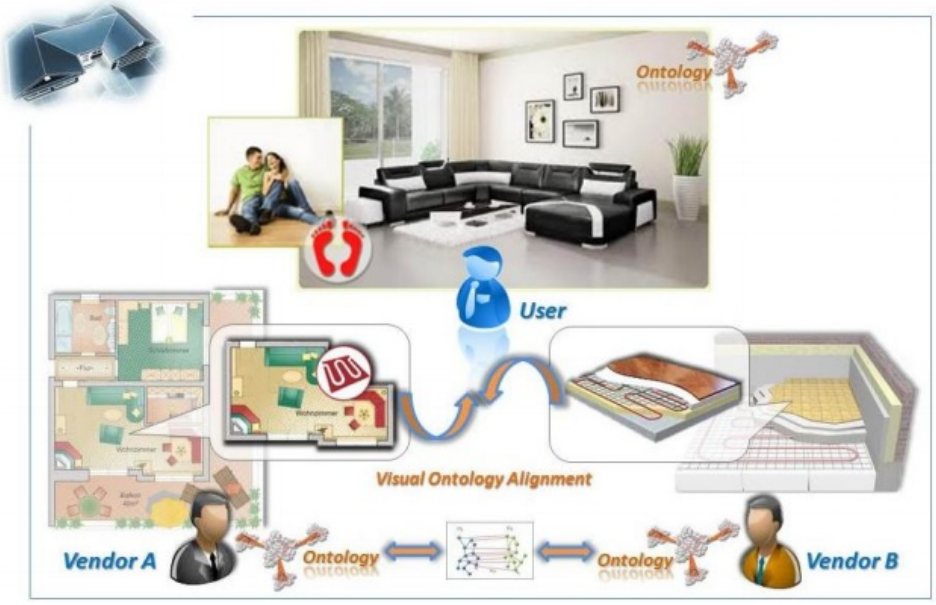
Esse capítulo apresenta trabalhos que tratam sobre o problema de interoperabilidade semântica e que serviram de base para esta proposta. Conforme apresentado em (KHRIYENKO; TERZIYAN; KAIKOVA, 2012) os autores descrevem um gateway semântico que utiliza uma ontologia de alinhamento, feita manualmente pelo usuário, para tratar o problema de interoperabilidade semântica. Essa ontologia de alinhamento é construída por meio de uma interface para alinhamento visual de ontologias que foi proposta pelos autores. O problema dessa abordagem é que exige necessariamente a intervenção do usuário.

A figura 3.1 ilustra um exemplo citado no trabalho. Dois fornecedores provêm sensores para pavimentos residenciais, cada um fornece uma ontologia para o seu produto. No exemplo, o usuário comprou os sensores dos dois fornecedores para colocar em sua casa. Existe também uma aplicação residencial que utiliza os dados desses sensores para prover um comportamento mais inteligente para a residência. Para que essa aplicação tenha acesso aos seus dados, ambos os sensores devem enviar suas informações para a aplicação, sendo que todas as mensagens sempre passarão pelo gateway semântico. No gateway semântico existe uma ontologia de alinhamento entre os dois padrões de ontologias dos sensores e o padrão de ontologia da aplicação que foi feita previamente pelo usuário. Por meio dessa ontologia, os conceitos mapeados pelo usuário são convertidos para serem compreendidos por ambas as partes.

Em (WANG et al., 2012) foi proposto uma modelagem semântica para solucionar o problema de interoperabilidade em IoT. O objetivo do trabalho não foi propor algum padrão, mas sim módulos de ontologias necessários para modelar toda a semântica de Internet das Coisas. Os módulos propostos pelos autores estão ilustrados na figura 3.2 Como é possível visualizar, é composta de sete módulos:

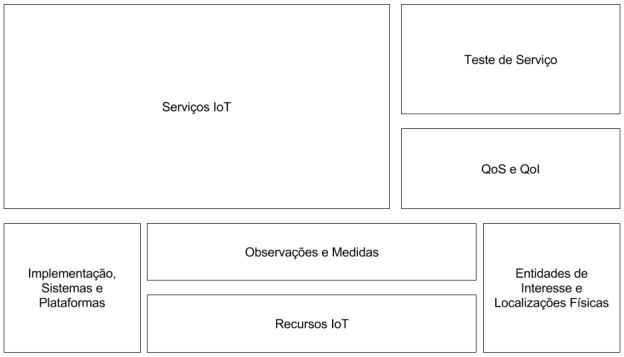
- *Recursos IoT*: Apresenta essencialmente a descrição de atuadores e sensores no ambiente IoT, podendo se estender também para descrever gateways IoT e servidores.
- *Serviços IoT*: Esse módulo modela os serviços, basicamente apresentam o contexto de uso e como acessá-lo.
- *QoS e QoI*: Modelam métricas e regras para qualidade de serviço e informaçao em IoT.
- *Teste de Serviço*: Modelam os testes e verificações das capacidades funcionais e não funcionais dos serviços IoT.

Figura 3.1: User assisted semantic interoperability in Internet of Things




Fonte: Imagem retirada de (KHRIYENKO; TERZIYAN; KAIKOVA, 2012)

Figura 3.2: Módulos de ontologias propostos pelos autores




Fonte: Imagem adaptada de (WANG et al., 2012)

- *Implementação, Sistemas e Plataformas*: Nesse módulo é descrito como os dispositivos estão organizados e suas relações.
- *Observações e Medidas*: Apresenta a representação semântica dos conceitos dos dados coletados pelos sensores.
- *Entidades de Interesse e Localizações Físicas*: Uma entidade de interesse de uma aplicação ou usuário e sua localização física estão bastante associadas, especialmente em service discovery, esse módulo visa descrever esses dois conceitos.

As duas principais diferenças desse conjunto de módulos apresentado por (WANG et al., 2012) para o proposto neste trabalho são:

- O conjunto de módulos proposto utiliza apenas parte dos módulos do trabalho de (WANG et al., 2012) que são relevantes no contexto delimitado pelo trabalho.
- A visão da modelagem dos autores em (WANG et al., 2012) é de todo o domínio, ou seja, considera uma visão geral de todos os dispositivos na rede IoT. Na proposta desse trabalho, será considerada uma visão parcial da rede, de apenas um dispositivo.

# 4 IDEIA CENTRAL

Como foi apresentado anteriormente, o problema de interoperabilidade de IoT não se limita a apenas a grande quantidade de protocolos entre dispositivos em uma única rede ou a grande diversidade de formatos que cada dispositivo fornece de suas observações, mas também a grande quantidade de silos[1] de informações que são criados a partir de Internet das Coisas. Fornecer a possibilidade de aplicações acessarem dispositivos de diversos silos aumentam as possibilidades de utilização da Internet das Coisas.

Este capítulo apresenta as principais ideias que foram utilizadas para uma possível solução do problema de interoperabilidade semântica entre aplicações em IoT. Como o trabalho visa a interoperabilidade em IoT, fazemos a suposição que todos os agentes estão nos padrões FIPA, garantindo a interoperabilidade sintática entre os agentes.

De modo geral, dispositivos e os serviços em IoT podem ser representados por uma ontologia e um agente. Gateways interligam redes de dispositivos, dessa forma é possível que mesmo que dois dispositivos que estejam em redes diferentes, possam se comunicar. Por razão da interoperabilidade semântica, além de se comunicar, é necessário que haja uma compreensão entre os dispositivos.

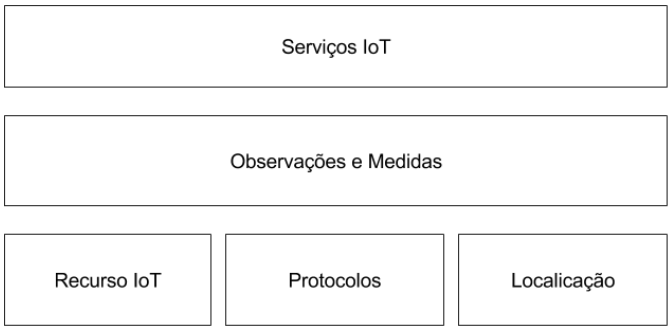
No primeiro subcapítulo são apresentados os módulos necessários de ontologia que representam um dispositivo e um serviço e o segundo subcapítulo apresenta os componentes que devem ser considerados para implementar um agente em IoT. Para prover algum serviço ao usuário esses agentes precisam se comunicar entre si, e muitas vezes um fornecedor não consegue prever todos os ambientes que esse dispositivo pode ser inserido. No terceiro subcapítulo é apresentado o agente tradutor. Esse agente é baseado no protocolo de coordenação blackboard, ficando entre sistemas de domínio diferentes. Ele utiliza alinhamento de ontologias para traduzir semanticamente as mensagens de um domínio para outro.

## 4.1 Conjuntos de ontologia

Esse trabalho utiliza um conjunto básico de ontologias para descrever um dispositivo IoT, baseado no conjunto encontrado em (WANG et al., 2012). Foi necessário analisar, a partir do conjunto inicial, quais grupos de ontologias eram relevantes para esse

[1] Referência à (SERRANO et al., 2015) que expõe que as aplicações em IoT hoje em dia são pensadas verticalmente e não horizontalmente (entre domínios)

Figura 4.1: Conjunto de ontologias proposto para o trabalho




Fonte: O Autor

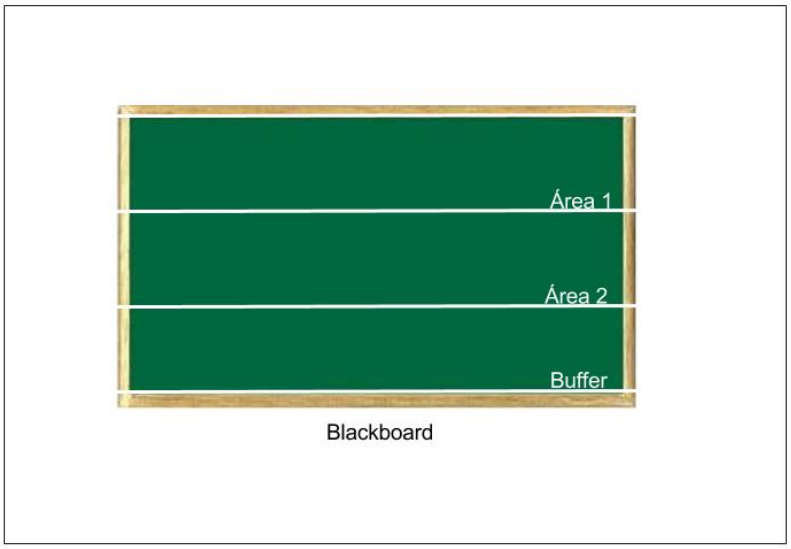
trabalho, ou seja, como o objetivo do trabalho é prover interoperabilidade semântica para os diferentes domínios em IoT, o conjunto "QoS e QoI" e "Teste de Serviço" foram suprimidos por estarem fortemente ligados à arquitetura SOA. Além disso, foi considerado que cada agente tem uma visão individual de si, não necessitando uma visão completa do domínio. Por essa razão, os conjuntos restantes sofreram algumas modificações.

Conforme a figura 4.1, são cinco conjuntos propostos:

- **Recurso IoT**: Nesse grupo são apresentadas as descrições físicas dos dispositivos IoT, sensores ou atuadores.
- **Protocolos**: São descritos nesse conjunto os protocolos de acesso aos serviços e ao dispositivo. Normalmente um protocolo é descrito através de entradas, saídas e conteúdo das mensagens.
- **Localização**: A localização descreve a área de atuação do atuador ou a área de persepção do sensor. Pode ser utilizado nas descobertas de serviços ou dispositivos por muitos nodos da rede.
- **Observações e Medidas**: Apresenta a descrição semântica dos dados coletados pelo dispositivo, é o modulo com foco principal neste trabalho, uma vez que é possível obter informações, desde que as entidades que descrevem essas informações sejam bem definidas semanticamente.
- **Serviços IoT**: Além da busca de dados, um dispositivo IoT pode buscar serviços na rede. Sendo assim, é necessário descrever semanticamente os serviços IoT.

A partir desses grupos de ontologia é possível implementar descoberta e interação entre agentes de domínios diferentes. Por exemplo, caso se deseje implementar descoberta de serviço, as ontologias dos módulos "Serviços IoT" e "Localização" (normalmente o

Figura 4.2: Imagem ilustrativa do agente tradutor




Fonte: O Autor

serviço tem um alcance, e ele é levado em consideração na descoberta de serviços) serão utilizadas, e para acessar o serviço, será necessário utilizar as ontologias do grupo "Protocolos".

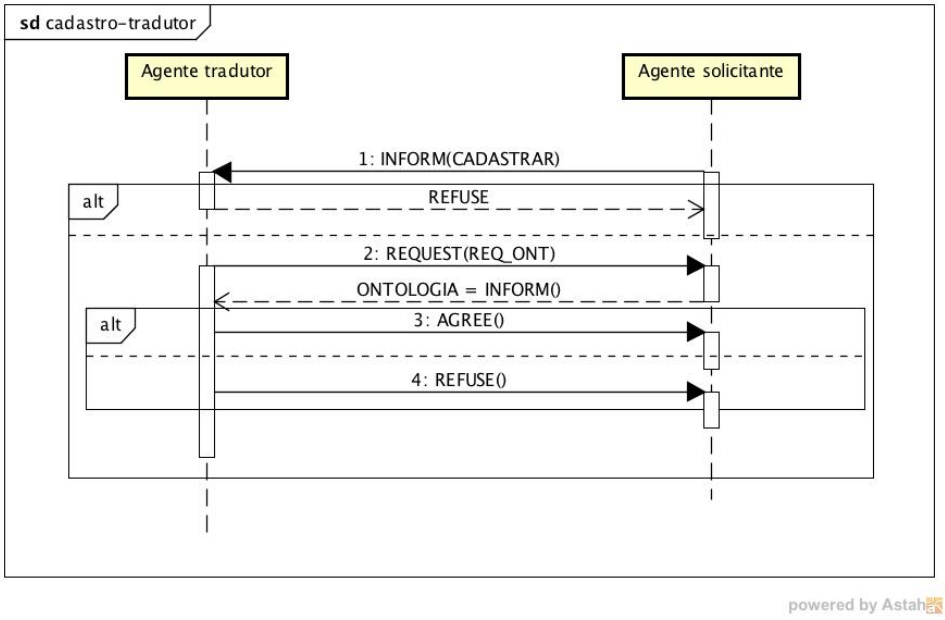
## 4.2 Agente Tradutor

O agente tradutor é um facilitador para a comunicação entre dois agentes de domínios diferentes, ou seja, que tem modelos de conhecimento diferentes. Nesse trabalho, foi decidido que o agente seguiria uma arquitetura blackboard, dividida em áreas, cada área correspondente para um padrão de ontologias e cada agente que segue determinado padrão de ontologias pode ler informações que são escritas em sua área e podem se comunicar com outros agentes de padrão de dados diferentes escrevendo em outras áreas.

Na figura 4.2 é possível notar que existem três áreas no agente tradutor dado como exemplo. As duas primeiras são usadas para troca de mensagens entre agentes, onde cada uma representa um padrão de ontologia. A última, é utilizada como um buffer, onde cada mensagem que chega passa antes por essa área, para realizar o processo de tradução e ser direcionada às outras áreas.

Para um agente utilizar o blackboard deve estar cadastrado antes no agente tradutor. Ao receber a solicitação de um cadastro, o agente tradutor verifica se para o padrão de ontologia do agente já existe uma área, caso a resposta seja positiva é informado apenas qual o quadro que deve ser utilizado pelo agente que solicitou o cadastro, se a resposta

Figura 4.3: Protocolo de cadastro no agente tradutor




Fonte: O Autor

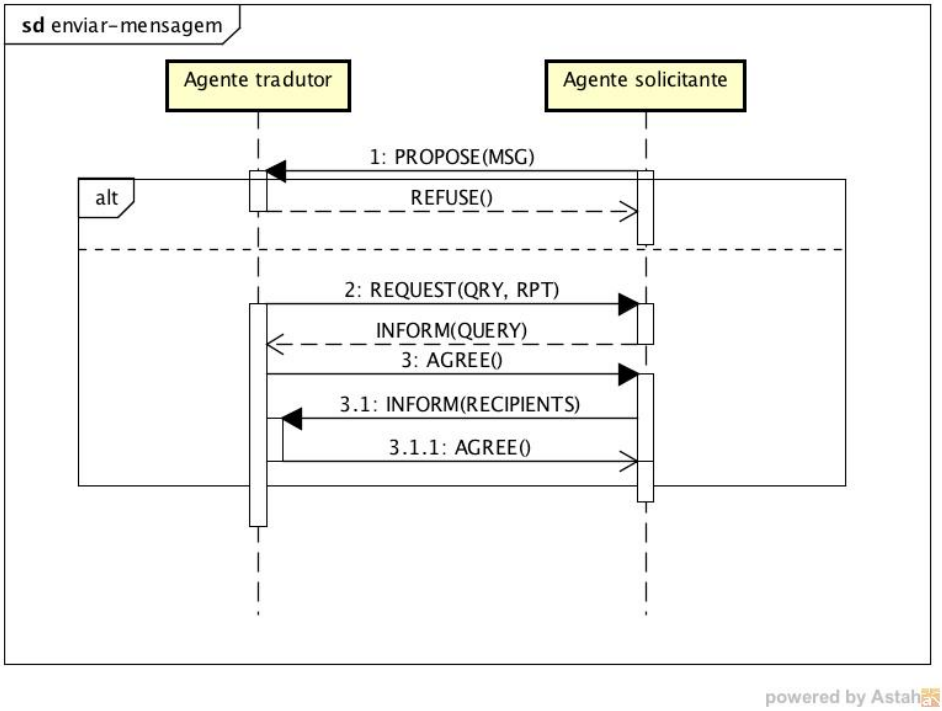
é negativa o agente tradutor deve criar uma área para o novo agente e posteriormente informar a área.

A figura 4.3 demonstra o protocolo para se cadastrar no agente tradutor de acordo com o padrão FIPA. Quem inicia o protocolo é sempre o agente que deseja se cadastrar no agente tradutor. O agente solicitante inicia a interação com uma mensagem do tipo INFORM, informando que deseja se cadastrar. O agente tradutor verifica se tem capacidade para gerir mais um agente, se não tiver ou o agente já estiver cadastrado o protocolo é encerrado com um REFUSE (e caso já esteja cadastrado, com o número da área correspondente do agente solicitante), caso contrário o agente tradutor solicita a ontologia do agente solicitante, com uma mensagem do tipo REQUEST. O solicitante responde com a URI da ontologia que utiliza. Caso o processo tenha ocorrido sem problemas, o agente tradutor responde com um AGREE com o identificador da área criada, caso contrário a resposta será REFUSE.

Após o cadastro, para postar uma mensagem um agente deve utilizar SPARQL, considerando o seu próprio padrão de ontologia, e informar para quais agentes essa mensagem será postada. O agente tradutor informará um identificador único para a mensagem, através dele poderá ser obtida a resposta da mensagem enviada.

Na figura 4.4 é mostrado o protocolo para um agente publicar sua mensagem no quadro. O agente solicitante inicia a comunicação com uma mensagem PROPOSE, informando seu desejo para postar a mensagem. Ao receber a mensagem, o agente tradutor

Figura 4.4: Protocolo de envio de mensagem para o agente tradutor




Fonte: O Autor

verifica se o agente solicitante já esta cadastrado, caso não esteja o agente tradutor recusa o solicitante, se já está cadastrado o agente tradutor cria um identificador para mensagem, informa esse identificador para o agente solicitante e espera a mensagem que deseja transmitir e para quem deseja transmitir. De maneira assíncrona, o agente solicitante informa a mensagem e para quem enviá-la. A cada recebimento de informação, o agente tradutor envia uma confirmação para o solicitante. Após o recebimento da mensagem ela é replicada para cada destinatário.

A arquitetura blackboard proposta é dirigida a eventos, ou seja, todas as vezes que uma tarefa for postada ou excluída de uma área do quadro, os agentes que seguem o quadro serão notificados da alteração em sua área. Isso permite que se uma tarefa foi postada em diversas camadas do blackboard, e um agente se comprometeu com a tarefa, todos os outros agentes devem ser notificados para não perderem tempo com a verificação da tarefa. Além disso, evita sobrecarregar o agente blackboard, com requisições dos agentes apenas para verificar suas áreas no quadro.

Como a área no blackboard pode ser compartilhada entre diversos agentes, é interessante que o agente possa filtrar as mensagens que são relevantes para ele. Para isso as mensagens do blackboard tem um conjunto de metadados que descrevem algumas características básicas de uma mensagem. A tabela 4.1 demonstra os metadados utilizados

para filtragem de mensagens nesse trabalho.

Tabela 4.1: Metadados de uma mensagem do agente tradutor

| *Metadado* | *Descrição* |
|---|---|
| Destinatário | Metadado que representa os destinatários da mensagem. O destinatário é definido por uma identificação única de cada agente. |
| Remetente | Metadado que representa o remetente da mensagem. Assim como o destinatário é representado pela identificação única de cada agente. |
| Identificador | Esse metadado é utilizado para filtrar a mensagem por identificador. Quando um agente recebe uma resposta de uma mensagem, por padrão a resposta terá o mesmo identificador. |
| Hora Recebimento | É possível filtrar as mensagens por horário de recebimento, dessa forma é possível ordená-las e os agentes avaliarem em fila as mensagens |
| Bloqueada | Dois agentes não podem avaliar a mesma mensagem ao mesmo tempo, dessa forma, quando um agente está utilizando uma mensagem, esta fica bloqueada. Com esse metadado é possível filtrar as mensagens que estão ou não bloqueadas. |
| Serviço | É possível definir para qual serviço do agente destinatário será endereçada essa mensagem. |

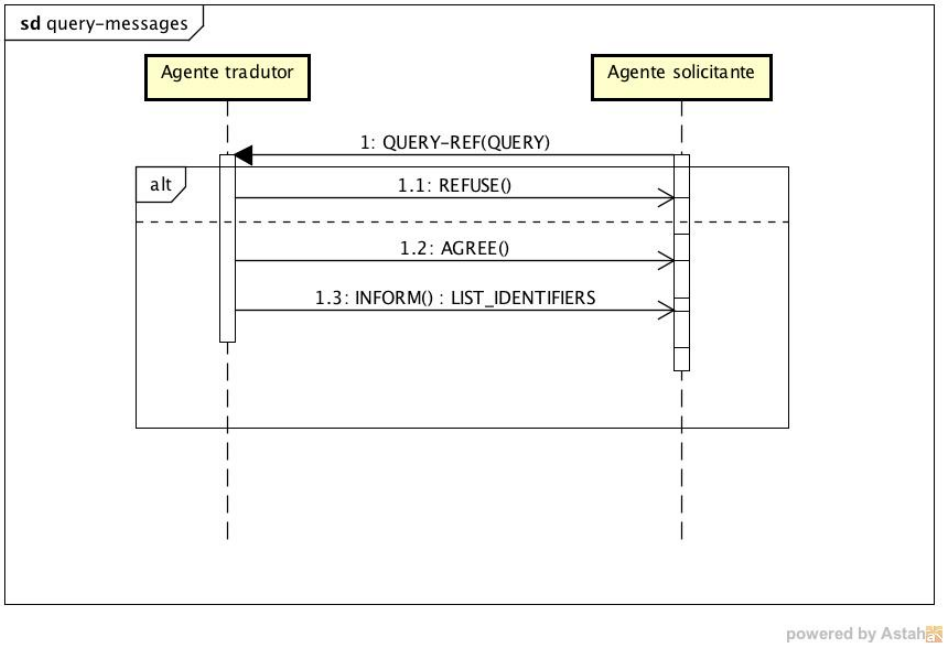
Fonte: O Autor

Os metadados são utilizados para filtrar as mensagens em forma de uma query. Nesse trabalho, foram utilizados operadores lógicos para complementar as queries. Na figura 4.5 é apresentado o protocolo para filtragem de mensagens no agente tradutor. O retorno do protocolo é um conjunto de identificadores de mensagens que satisfazem a query do agente solicitante, trazendo por ordem da mais antiga para a mais atual.

O metadado *Serviço* é utilizado para o agente destinatário saber qual serviço está sendo invocado pelo agente remetente. Como foi visto anteriormente, os serviços são representados utilizando o padrão OWL-S, onde se pode representar serviços de forma atômica ou composta. Em caso de representação composta, o agente tradutor deve guardar a etapa que está na comunicação. Quando o agente destinatário receber uma mensagem com o mesmo identificador deve passar para a próxima etapa do serviço. Para simplificar a implementação, nesse trabalho consideramos apenas serviços em forma atômica.

Para ler uma mensagem específica, o agente precisa ter o identificador da mensagem (que normalmente é obtida através do processo de filtragem). O agente solicitante usa uma mensagem INFORM para informar qual mensagem deseja ler. Se a mensagem

Figura 4.5: Protocolo para filtrar mensagens no agente tradutor




Fonte: O Autor

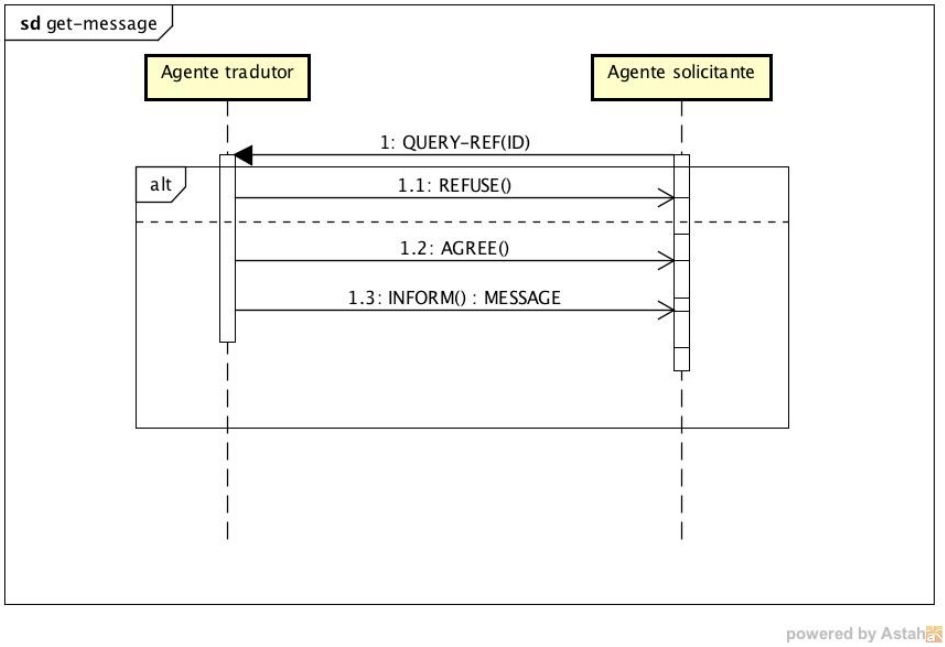
não estiver bloqueada e existir na área do quadro do agente solicitante, o agente tradutor bloqueia a mensagem e envia para o agente solicitante, caso contrário envia uma mensagem do tipo REFUSE. Esse protocolo é possível observar na figura 4.6.

## 4.3 Agente de Alinhamento

Na figura 4.2 existe uma área de buffer onde chegam todas as mensagens antes de serem traduzidas para outros padrões.O processo de alinhamento de ontologias pode prejudicar o desempenho do sistema, tempo de resposta do agente tradutor e aumentar o tamanho da área do buffer. Para evitar isso o agente de alinhamento foi criado. Ele é o responsável pelo mapeamento das ontologias, tradução das mensagens e de alguns metadados dela. Quando um agente se cadastra no tradutor, o agente de alinhamento recebe uma notificação para fazer o alinhamento da nova ontologia com as ontologias já existentes dos outros agentes já cadastrados e todas as vezes que uma nova mensagem chega, ele é acionado para traduzir a mensagem.

Uma vez que o agente de alinhamento recebe a notificação que existe uma nova mensagem em sua área no agente tradutor, a mensagem é processada pelo agente e verificada para quais camadas deve ser replicada. Para cada replicação é necessário utilizar um mapeamento dos conceitos do esquema de ontologia do remetente para os conceitos do

Figura 4.6: Protocolo para ler uma mensagem no agente tradutor



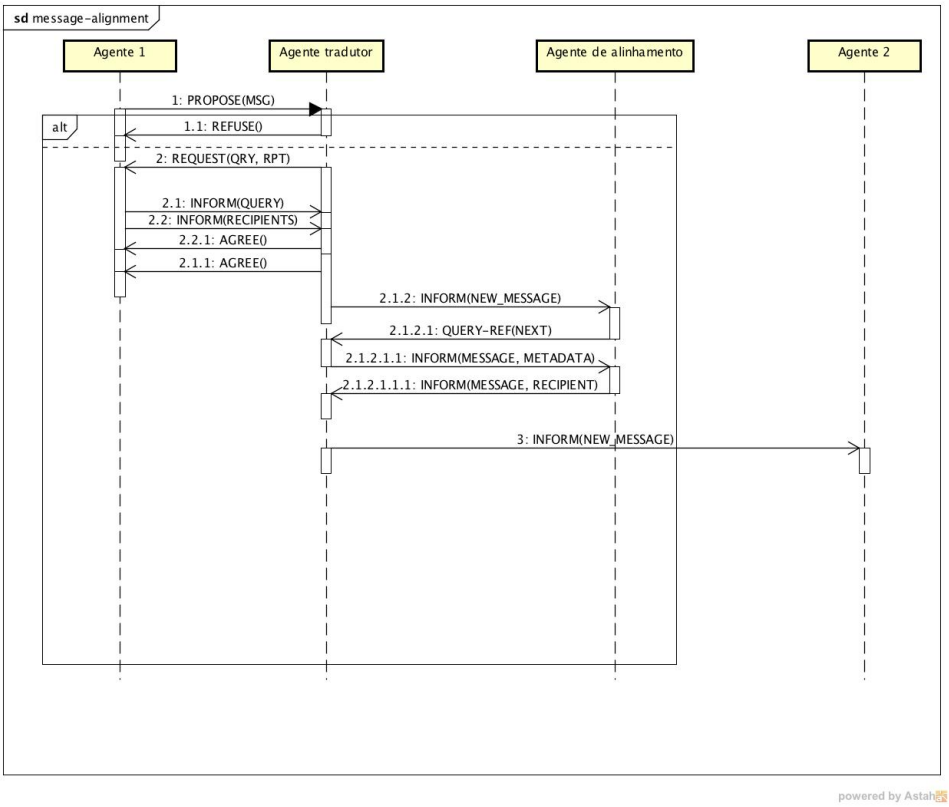

Fonte: O Autor

esquema de ontologia do destinatário, esse mapeamento é encontrado na ontologia de alinhamento, que mantém os relacionamentos entre os conceitos próprios de cada ontologia e relaciona os conceitos que foram alinhados das duas ontologias. Caso não exista previamente essa ontologia, o remetente deve ser notificado por um REFUSE, pois isso significa que o agente não está cadastrado ou não tem um mapeamento com o agente destinatário. Caso contrário a mensagem é postada na camada e os agentes dela são notificados.

A figura 4.7 ilustra o que foi descrito no parágrafo anterior. O Agente 1 envia uma mensagem para o agente tradutor, imediatamente o agente de alinhamento é notificado com um INFORM o recebimento de mensagens. O agente de alinhamento, se não estiver já ocupado, envia uma mensagem QUERY-REF para retirar a próxima mensagem do quadro. A mensagem é retornada pelo agente tradutor, junto com os metadados dela (descritos na tabela 4.1). Para cada tripla <mensagem, remetente, destinatário>, o agente de alinhamento realiza uma tradução e uma postagem no agente tradutor. Após conseguir traduzir os conceitos do agente 1 para o agente 2, posta a mensagem na camada do agente 2, que é notificado do recebimento da mensagem. Esse fluxo se torna bastante importante em IoT, especialmente em descoberta de serviços. Essa arquitetura permite que, mesmo que os domínios sejam diferentes, os dispositivos possam buscar serviços em comum.

O agente de alinhamento também é notificado quando um agente se cadastra no agente tradutor, para criar a ontologia de alinhamento, realizando o alinhamento e merge

Figura 4.7: Protocolo para alinhamento de mensagem




Fonte: O Autor

das ontologias do novo agente com as já existentes. Dessa maneira todas as vezes que mensagens são trocadas entre agentes de diversas áreas, não é necessário refazer o alinhamento todo novamente, apenas é necessário converter a mensagem.

Nesse trabalho o algoritmo de similaridade escolhido para o alinhamento foi o Tri-Gram, que é uma variação do N-Gram. O N-Gram clássico, ao comparar duas palavras, cria um conjunto de todas as substrings de tamanho n para cada palavra. O cálculo de similaridade é dado pelo número de substrings em comum das palavras, dividido pelo número de substrings do maior conjunto. O único parâmetro que muda do Tri-Gram para o N-Gram é que o tamanho da substring no Tri-Gram é três.

A aplicação do algoritmo de similaridade em um conceito é feita para todos os conceitos da outra ontologia. Depois dessa etapa, é possível determinar o grau de similaridade entre todos os conceitos, selecionando para cada conceito um par com o maior grau de similaridade entre todos e diferente de zero. Alguns conceitos podem ficar sem par, entretanto pode ser aceitável, uma vez que podem existir conceitos que são representados em algumas ontologias e abstraídos em outras.

Cada par de conceitos deve ter uma relação na ontologia de alinhamento, que em OWL será representada através das relações EquivalentProperty (utilizada para proprieda-

des de dados ou objetos equivalentes) e EquivalentClass (utilizada para classes equivalentes). Entretanto, nesse processo podem ser inseridas algumas inconsistências semânticas. Por exemplo, um algoritmo de similaridade pode mapear uma classe de uma ontologia para duas outras classes classes não-equivalentes da outra ontologia. É necessário que haja um processo de teste para procurar possíveis inconsistências geradas pelo merge das ontologias. Esse processo é realizado pelo reasoner. Caso seja encontrado algum problema, há necessidade de intervenção humana para solucioná-lo.

# 5 IMPLEMENTAÇÃO

A metodologia de desenvolvimento utilizada para a implementação do sistema de impressão multiagente pode ser encontrada em (NIKRAZ GIOVANNI CAIRE, 2006), foi utilizado o framework JADE para simplificar o desenvolvimento, além disso esse framework implementa o padrão FIPA para o desenvolvimento de agentes. A metodologia é especifica para sistemas multiagentes e se divide em três grandes etapas que são análise, desenvolvimento e implementação. Não será mostrado o resultado de cada etapa nesse trabalho, entretanto ao longo desse capítulo serão mostrados alguns artefatos resultantes de cada uma delas.

Esse capítulo se divide em duas grandes sessões. Na primeira sessão, será mostrado como foi planejado o estudo de caso para testar os agentes propostos, a descrição do ambiente, os agentes que compõem o ambiente e como serão usados o agente tradutor e o agente de alinhamento para resolver o problema de impressão. A segunda sessão demonstra os resultados da aplicação do estudo de caso.

## 5.1 Estudo de caso

No estudo de caso, vamos avaliar os agentes propostos no contexto de um sistema de impressão observando seu comportamento quando exposto a situações onde deve ser consideradas diversas impressoras, cada uma com uma representação diferente do seu ambiente e com funcionalidades diferentes. Apesar de não ser um exemplo literalmente imerso no ambiente IoT, ilustra os problemas que são frequentemente encontrados em IoT, como descoberta de dispositivos e serviços. A ideia principal é implementar um agente para cada impressora, um agente para o servidor e um para cada usuário e realizar suas interações de forma que o agente blackboard e os agentes reasoners possam prover interoperabilidade semântica para o sistema.

No primeiro subcapítulo é feita uma descrição do ambiente utilizado nesse caso de estudo, como impressoras utilizadas, protocolos, computadores e usuários. Após a descrição do ambiente são apresentados as modelagens de ontologias consideradas para cada impressora. No terceiro subcapítulo é apresentada a metodologia de desenvolvimento dos agentes. Por fim, são apresentados os testes realizados e os resultados.

### 5.1.1 Descrição do ambiente

Para o estudo de caso foi implementado um ambiente artificial de uma universidade, considerando impressoras (com modelos e serviços diferentes), computadores e permissões de usuário.

Tabela 5.1: Impressoras do ambiente de teste

| *Modelo* | *Fabricante* | *Permissões* |
|---|---|---|
| CM1415 | HP | Bolsistas IC ou Mestrado |
| HP LaserJet P4015 | HP | Todos os alunos e professores do INF |
| L565 | Epson | Todos os alunos |
| CX310dn | Lexmark | Todos os professores |

Fonte: O Autor

É possível observar a presença de quatro impressoras no ambiente descrito na tabela 5.1. Todos os usuários podem usar ao menos uma das impressoras, e é garantido que todos os serviços são acessíveis a eles. Abaixo são descritas as principais informações sobre as impressoras. Foram consideradas apenas informações relevantes para esse trabalho e que depois serão formalizadas através de ontologias:

- **CM1415**: A primeira impressora é da HP, modelo CM1415, e como descrito acima, tem permissão de uso de bolsistas IC e alunos de mestrado. Essa impressora é uma multifuncional à laser, imprime e copia até 12 paginas por minuto monocromáticas e 8 páginas por minuto coloridas, com uma resolução de até 300 dpi, tem 128MB de memória RAM e impressão frente e verso manual.
- **P4015**: A segunda impressora, também é da HP, modelo HP LaserJet P4015, da permissão de uso para todos os alunos e professores do INF. É apenas uma impressora monocromática, entretanto imprime e copia até 52 páginas por minuto, resolução de 600dpi, 128MB de memória RAM e impressão frente e verso automática.
- **L565**: A impressora seguinte é da Epson, modelo L565, todos os alunos da UFRGS tem permissão para imprimir nessa impressora. Imprime e copia 9,2 paginas por minuto monocromáticas e 4,5 páginas por minuto coloridas, resolução de 300dpi, a memória interna não é informada (nesse caso, o agente que controla a impressora considera apenas um job por vez) e impressão frente e verso manual.
- **CX310dn**: A ultima impressora é uma Lexmark, modelo CX310dn, onde todos os professores da UFRGS podem acessa-la. Tem velocidade de impressão e copia de até 23 páginas por minuto, tanto para monocromática como para colorida. Tem resolução de até 1200 dpi e impressão frente e verso automática. A memória

considerada é 512MB.

Tabela 5.2: Computadores do ambiente de teste

| *Nome* | *Agentes* |
|---|---|
| Computador 1 | Agentes impressoras CM1415, HP LaserJet P4015, L565 e CX410de |
| Computador 2 | Agente servidor |
| Computador 3 | Agente blackboard e reasoners |
| Computador 4 | Agentes usuários |

Fonte: O Autor

A tabela 5.2 apresenta os computadores utilizados para rodar os agentes. No ambiente de teste foi utilizado 4 computadores, todos conectados à mesma rede, e os agentes estão separados por tipo, sendo que não há nenhuma limitação de comunicação entre os agentes.

Tabela 5.3: Permissões do ambiente de teste

| *Nome* | *Descrição* |
|---|---|
| Bolsistas IC | Nesse grupo são considerados os bolsistas de iniciação cientifica do instituto de informática |
| Alunos de Mestrado | Nesse grupo são considerados os alunos de mestrado do instituto de informática |
| Alunos do INF | Nesse grupo são considerados todos os alunos do instituto de informática |
| Alunos da UFRGS | Nesse grupo são considerados todos os alunos da UFRGS, incluindo os alunos do instituto de informatica |
| Professores da UFRGS | Nesse grupo é considerado o corpo docente da UFRGS. |
| Professores do INF | Nesse grupo é considerado o corpo docente do instituto de informática |

Fonte: O Autor

Além das impressoras e dos computadores, serão considerados alguns usuários padrões que irão ter algumas das permissões da tabela 5.3, e serão utilizados como exemplo para este estudo de caso:

- **João Silva**: Um professor INF que utiliza o computador de sua sala para utilizar o serviço de impressão. (dizer depois onde fica localizada a sala do professor e as impressoras)
- **Mateus Cardoso**: É aluno UFRGS, mas não pertence ao instituto de informática. Utiliza seu notebook para acessar a rede e utilizar o serviço de impressão.
- **Evelyn Oliveira**: É uma professora da UFRGS, mas também não vinculada ao instituto de informática, acessa o serviço de impressão de seu celular.

- **Brenda Barbosa**: Aluna vinculada ao instituto de informática, utiliza o laboratório de computadores para acessar a rede e utilizar o serviço de impressão.

### 5.1.2 Descrição dos conjuntos de ontologia

Este subcapítulo será apresentado por padrões, serão apresentados os módulos de ontologia de cada padrão de impressora, de servidores e de usuários. Considerando as impressoras, devemos propor 3 padrões de ontologias: Um padrão para as impressoras HP (CM1415 e P4015), um para Epson e um para Lexmark, dessa forma impressoras de diferentes fornecedores tem visões diferentes do mesmo domínio. O outro domínio é o do servidor, que vai prover o serviço de impressão, para essa entidade também é necessário um padrão para os módulos de ontologia. Por fim, o outro domínio é o do usuário, que nesse caso, busca um serviço de impressão.

Para simplificar a modelagem das ontologias, foi considerado o conjunto de localização o mesmo para todos os padrões. Foi utilizada a ontologia GML v.1, descrita na tabela 5.4 para representar a localização dos dispositivos no estudo de caso.

Tabela 5.4: Ontologia GML v.1

| *Classe* | *Propriedades de dados* |
|---|---|
| Geometry | latitude |
| Point | longitude |

Fonte: O Autor

No padrão GML v.1, o namespace é *http://gsw.projects.semwebcentral.org/2005/03/geofilter/gml-ont*. Nessa ontologia é possível visualizar as seguintes classes:

- *Geometry* : É a classe que pode representar um ponto, uma linha ou uma área, no caso deste trabalho, consideramos apenas um ponto.
- *Point* : É subclasse da classe Geometry e representa um ponto na coordenada geográfica. As propriedades de dados dessa classe são:
  - *latitude* : É um dado tipo float, que representa a latitude do ponto.
  - *longitude* : É um dado tipo float, que representa a longitude do ponto.

O padrão HP, descrito na tabela 5.5, tem como namespace *http://hp.com/ontologia/impressora*. Todos os elementos das ontologias são referenciados por *namespace#Nome do elemento*. No módulo de RecursosIoT, é possível visualizar os seguintes elementos:

- *HPImpressora*: A classe que descreve a impressora, características físicas e princi-

Tabela 5.5: Padrão HP

| *Conjunto* | *Classe* | *Propriedades de dados* | *Propriedades de objetos* |
|---|---|---|---|
| Recursos IoT | HPImpressora | modeloImpressora | |
| | | descricaoImpressora | |
| | | tipoDuplex | |
| | | memoriaRAM | |
| | | capacidadePapel | |
| | HPCartucho | modeloCartucho | |
| | | tipoImpressao | |
| | | tipoTinta | |
| Protocolos | HPProtocolo | nomeProtocolo | |
| | | descricaoProtocolo | |
| | HPMensagem | Performative | |
| | | Conteudo | |
| | | Receiver | |
| | | Step | |
| Observações e Medidas | HPMedidas | nomeMedida | |
| | | valorMedida | |
| Serviços IoT | HPServicos | nomeServico | hasPrecondition |
| | | descricaoServico | hasPoscondition |

Fonte: O Autor

palmente descreve o que é o produto. Por isso, essa classe tem as seguintes propriedades de dados:

- *modeloImpressora*: Modelo da impressora, propriedade usada para apresentar ao usuário em qual impressora a tarefa esta sendo impressa.
- *descricaoImpressora*: Descrição da impressora pelo fabricante.
- *tipoDuplex*: São possíveis dois valores “automático”, caso a impressora tenha essa funcionalidade de imprimir dos dois lados automática, ou “manual”, caso essa funcionalidade seja manual.
- *memoriaRAM*: Quantidade de memória RAM em MB da impressora. Essa propriedade será utilizada pelo agente da própria impressora para controlar quantas tarefas pode mandar para o dispositivo.
- *capacidadePapel*: Quantidade máxima de folhas que a bandeja da impressora suporta. Essa propriedade será utilizada pelo agente da própria impressora para controlar suas tarefas.

- *HPCartucho*: Essa classe representa um cartucho/tonner da impressora. Com essa representação é possível saber com quais tipos de tinta a impressora pode imprimir. Por exemplo, é possível que em uma impressora multifuncional, que a impressão utilize os dois cartuchos (monocromático e colorido), dessa forma a impressão

presta o serviço colorido e monocromático. As propriedades de dados que essa classe tem são:

- *modeloCartucho*: Modelo do cartucho/tonner da impressora
- *tipoImpressao*: Os valores possíveis para essa propriedade é "laser" caso seja impressão a laser ou "tinta" caso seja impressão a tinta.
- *tipoTinta*: São possíveis dois valores, "monocromático" caso o cartucho seja monocromático, ou "colorido" caso seja um cartucho colorido.

O módulo de protocolos, assim como o módulo anterior, também é composto por duas classes. A descrição de cada uma delas com as propriedades de dados e objetos segue abaixo:

- *HPProtocolo*: Essa classe representa um protocolo do agente de uma impressora HP. Utilizado para acessar os serviços que o agente provê. As propriedades de dados relacionadas a essa classe são:
  - *nomeProtocolo*: Nome do protocolo
  - *descricaoProtocolo*: Descrição escrita do protocolo para fins de conhecimento pelo usuário.
- *HPMensagem*: Representa uma mensagem de um protocolo. As propriedades de dados pertencentes a essa classe são:
  - *performative*: Considerando que os agentes seguem o padrão FIPA, esse campo é uma performativa FIPA da mensagem.
  - *conteudo*: Identifica qual tipo de conteúdo da mensagem, pode ser SPARQL, texto, ou outra que o agente suporta.
  - *receiver*: O receiver identifica para qual agente a mensagem deve ser enviada.
  - *step*: Representa o passo do protocolo, caso haja uma sequência de mensagens trocadas.

O conjunto de observações e medidas apresenta apenas uma classe:

- *HPMedidas*: Essa classe representa os dados coletados pela impressora. As propriedades de dados que representam esses dados coletados são:
  - *nomeMedida*: Essa propriedade determina o nome do dado coletado.
  - *valorMedida*: Essa propriedade determina o valor do dado coletado.

O último conjunto no padrão HP é o Serviços IoT:

- *HPServico*: Classe que determina um serviço prestado pela impressora. As propriedades de dados que são relacionados a essa classe são:
  - *nomeServico*: Essa propriedade determina o nome do serviço.
  - *descricaoServico*: Essa propriedade determina a descrição do serviço.

  Além das propriedades de dados, essa classe também tem propriedade de objetos:
  - *hasPrecondition*: Relaciona a classe HPServico com alguma pré-condição necessária para executar um serviço.
  - *hasPostcondition*: Relaciona a classe HPServico com alguma pós-condição do serviço.

Tabela 5.6: Padrão Epson

| *Conjunto* | *Classe* | *Propriedades de dados* | *Propriedades de objetos* |
|---|---|---|---|
| Recursos IoT | EpsonImpressora | modeloImpressora | |
| | | descricao_impressora | |
| | | tipo_duplex | |
| | | memoria_ram | |
| | | capacidade_papel | |
| | | tipo_tinta | |
| Protocolos | EpsonProtocolo | nome_protocolo | |
| | | descricao_protocolo | |
| | EpsonMessagem | performative | |
| | | content | |
| | | receiver | |
| | | step | |
| Observações e Medidas | EpsonMedidas | nome_medida | |
| | | valor_medida | |
| Serviços IoT | EpsonServico | nome_servico | hasPrecondition |
| | | descricao_servico | hasPoscondition |

Fonte: O Autor

O padrão da Epson está descrito na tabela 5.6 com suas clases e propriedades de dados e objetos. Ele tem como namespace *http://epson.com/ontologia/impressora* e o nome dos elementos da ontologia seguem o mesmo padrão da ontologia anterior. O primeiro conjunto é de Recurso IoT:

- *EpsonImpressora*: A classe que descreve a impressora, características físicas e principalmente descreve o que é o produto. Por isso, essa classe tem as seguintes propriedades de dados:
  - *modeloImpressora*: Modelo da impressora, propriedade usada para apresentar ao usuário em qual impressora a tarefa esta sendo impressa.

- *descricao_impressora*: Descrição da impressora pelo fabricante.
- *tipo_duplex*: São possíveis dois valores "automático", caso a impressora tenha essa funcionalidade de imprimir dos dois lados automática, ou "manual", caso essa funcionalidade seja manual.
- *memoria_ram*: Quantidade de memória RAM em MB da impressora. Essa propriedade será utilizada pelo agente da própria impressora para controlar quantas tarefas pode mandar para o dispositivo.
- *capacidade_papel*: Quantidade máxima de folhas que a bandeja da impressora suporta. Essa propriedade será utilizada pelo agente da própria impressora para controlar suas tarefas.
- *tipo_tinta*: São possíveis dois valores, "monocromático" caso o cartucho seja monocromático, ou "colorido" caso seja um cartucho colorido.

O módulo de protocolos, assim como o módulo anterior, também é composto por duas classes. A descrição de cada uma delas com as propriedades de dados e objetos segue abaixo:

- *EpsonProtocolo*: Essa classe representa um protocolo do agente de uma impressora Epson. Utilizado para acessar os serviços que o agente prove. As propriedades de dados relacionadas a essa classe são:
  - *nome_protocolo*: Nome do protocolo
  - *descricao_protocolo*: Descrição escrita do protocolo para fins de conhecimento pelo usuário.
- *EpsonMensagem*: Representa uma mensagem de um protocolo. As propriedades de dados pertencentes a essa classe são:
  - *performative*: Considerando que os agentes seguem o padrão FIPA, esse campo é uma performativa FIPA da mensagem.
  - *content*: Identifica qual tipo de conteúdo da mensagem, pode ser SPARQL, texto, ou outra que o agente suporta.
  - *receiver*: O receiver identifica para qual agente a mensagem deve ser enviada.
  - *step*: Representa o passo do protocolo, caso haja uma sequência de mensagens trocadas.

O conjunto de observações e medidas apresenta apenas uma classe:

- *EpsonMedidas*: Essa classe representa os dados coletados pela impressora. As propriedades de dados que representam esses dados coletados são:
  - *nome_medida*: Essa propriedade determina o nome do dado coletado.
  - *valor_medida*: Essa propriedade determina o valor do dado coletado.

  O último conjunto no padrão Epson é o Serviços IoT:

- *EpsonServico*: Classe que determina um serviço prestado pela impressora. As propriedades de dados que são relacionados a essa classe são:
  - *nome_servico*: Essa propriedade determina o nome do serviço.
  - *descricao_servico*: Essa propriedade determina a descrição do serviço.

  Além das propriedades de dados, essa classe também tem propriedade de objetos:
  - *hasPrecondition*: Relaciona a classe EpsonServico com alguma pré-condição necessária para executar um serviço.
  - *hasPostcondition*: Relaciona a classe EpsonServico com alguma pós-condição do serviço.

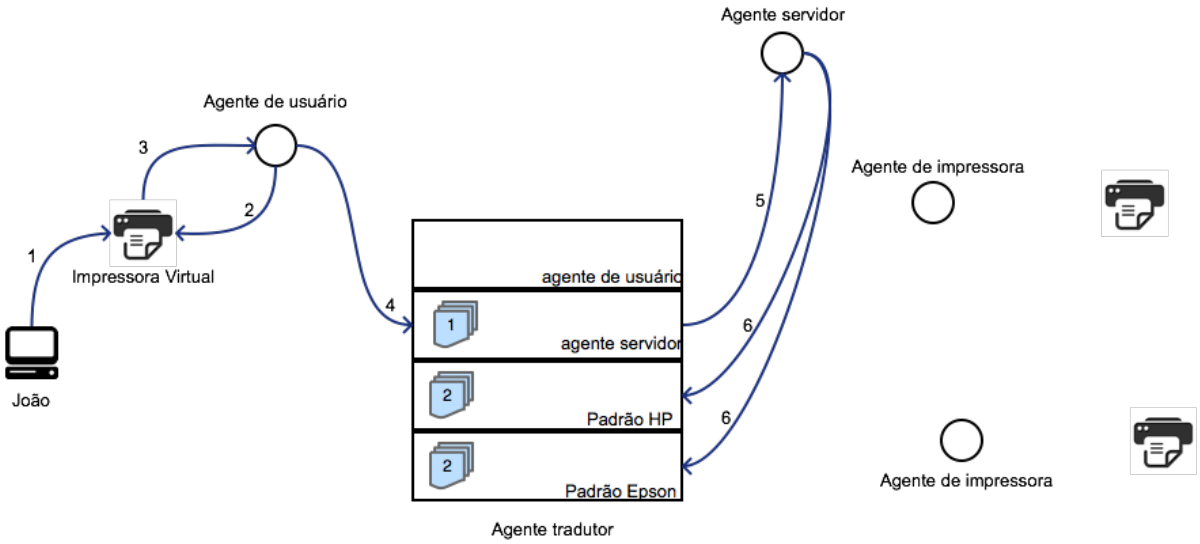
Tabela 5.7: Padrão Lexmark

| *Conjunto* | *Classe* | *Propriedades de dados* | *Propriedades de objetos* |
|---|---|---|---|
| Recursos IoT | ImpressoraLexmark | modeloimpressora | |
| | | descricaoimpressora | |
| | | tipoduplex | |
| | | memoriaram | |
| | | capacidadepapel | |
| | | tipotinta | |
| Protocolos | ProtocoloLexmark | nome_protocolo | |
| | | descricao_protocolo | |
| | MessagemLexmark | performativa | |
| | | conteudo | |
| | | receiver | |
| | | step | |
| Observações e Medidas | MedidasLexmark | nomemedida | |
| | | valormedida | |
| Serviços IoT | ServicoLexmark | nomeservico | hasPrecondition |
| | | descricaoservico | hasPoscondition |

Fonte: O Autor

O padrão Lexmark, descrito na tabela 5.7, é bastante similar com o padrão Epson. A única mudança ocorre no padrão utilizado para os nomes. Além disso, o namespace do padrão Lexmark é *http://lexmark.com/ontologia/impressora*.

Figura 5.1: Interação entre agentes - Passo 1



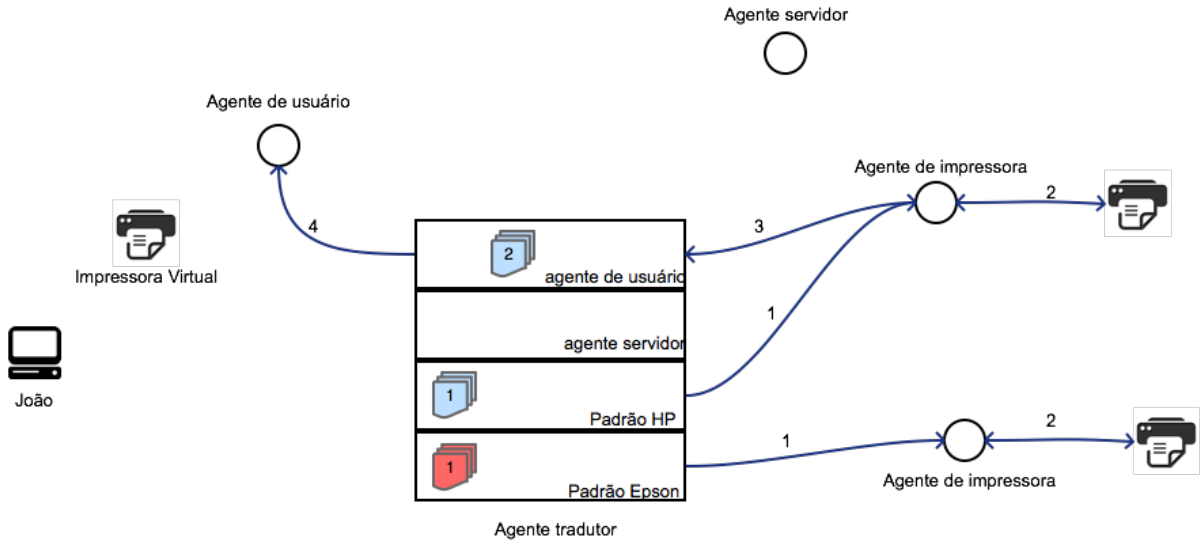

Fonte: O Autor

### 5.1.3 Interação entre os agentes

Como descrito anteriormente, além do agente tradutor e de alinhamento, no ambiente existem três tipos de agentes: agente usuário, agente impressora e o agente servidor. Antes de descrever cada agente, será explicado o funcionamento do sistema através das interações entre esses agentes. Foi elaborado um cenário para ilustrar o comportamento do sistema e as interações entre os agentes. O exemplo demonstrado considera um usuário chamado João, um agente servidor e dois agentes impressoras, um representando a impressora CM1415 e outro representando a impressora L565. Na imagem é desconsiderado o agente de alinhamento para facilitar a visualização, mas como regra geral, toda a mensagem o agente tradutor recebe, irá para uma área de buffer que após processada pelo agente de alinhamento é encaminhada para cada área destino.

Como um passo anterior a esse cenário, todos os agentes devem estar cadastrados no agente tradutor. Na figura 5.1, quando João solicita a impressão, a única impressora que o computador dele tem disponibilidade para escolher localmente é a virtual criada pelo agente usuário que está sendo executado no computador de João. O agente usuário verifica constantemente novos trabalhos na fila da impressora virtual. Quando chega um novo trabalho, o agente usuário processa os requerimentos para o trabalho, anota os da-

Figura 5.2: Interação entre agentes - Passo 2




Fonte: O Autor

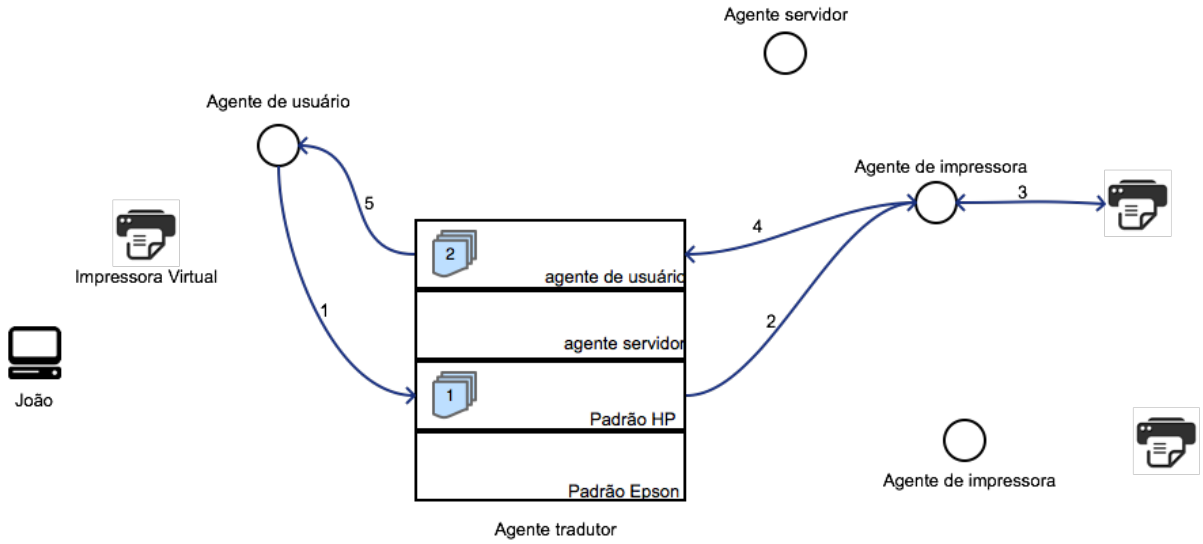
dos semânticamente os metadados e o conteúdo da mensagem, e envia uma requisição ao agente tradutor endereçando-a ao agente servidor para saber alguma impressora que satisfaça os requisitos de impressão solicitados.

A mensagem enviada pelo agente usuário é encaminhada inicialmente para o buffer do agente tradutor, que avisa o agente de alinhamento do recebimento de uma nova mensagem. O agente de alinhamento recebe a mensagem e verifica que o destinatário dessa mensagem é o agente servidor e traduz ela para o padrão dele. Na figura 5.1 a mensagem traduzida pode ser vista na área do agente servidor.

Quando o agente servidor recebe a mensagem em sua área no agente tradutor, ele é notificado e verifica que é uma consulta SPARQL endereçada para impressoras. Antes do agente servidor repassar essa tarefa para elas, é feita uma verificação de permissão do usuário, para saber quais impressoras ele tem direto de acessar. De acordo com o exemplo, João tem as duas impressoras do cenário(CM1415 e L565) como elegíveis. Selecionadas as impressoras, o agente servidor despacha uma mensagem para cada impressora elegível para impressão.

O agente de alinhamento novamente realiza a tradução da mensagem, dessa vez do padrão do servidor para cada uma das impressoras elegíveis para o usuário. Na figura 5.1 é possível visualizar uma mensagem em cada área dos agentes de impressora. Os

Figura 5.3: Interação entre agentes - Passo 3




Fonte: O Autor

agentes impressora competem entre si para acessar primeiro a mensagem em sua área. Na figura 5.2, a impressora CM1415 conseguiu acessar a mensagem antes da impressora L565, nesse caso, quando a impressora L565 for tentar acessar a mensagem, ela estará bloqueada pois já existe um agente utilizando a mensagem.

O agente da impressora CM1415 verifica que tem o serviço solicitado pelo usuário e retorna para o agente usuário e retorna ao agente usuário, como é mostrado na figura 5.3. Dessa forma, o agente usuário utiliza o serviço selecionado para enviar a tarefa de impressão para a impressora, enviando as informações necessárias (de acordo com o que está descrito na ontologia).

### 5.1.4 Agente Impressora

A proposta desse subcapítulo é descrever os agentes impressoras presentes no ambiente de estudo de caso. Existem dois pontos importantes na implementação dos agentes impressora, o primeiro é a comunicação com a impressora que se preocupa em qual protocolo utilizar para se comunicar com a impressora, como capturar as informações dos sensores da impressora (quantidade de tinta e quantidade de papel) e como enviar uma tarefa para ela. O segundo ponto é a comunicação com o agente tradutor, que além de im-

plementar o protocolo, deve ter representada em sua ontologia o fluxo de interação com ele.

A comunicação com impressora pode se dar, nesse trabalho, através de dois protocolos SNMP e IPP. O protocolo SNMP é um protocolo de gerenciamento de redes, que é utilizado para obter acesso às informações de nível de tinta e contador de páginas das impressoras que tem suporte a esse protocolo. Para acessar essas informações, o agente de impressora precisa ter conhecimento do OID (Object Identifier) do dado que deseja. Cada impressora tem um MiB (Management Information Base, que é o conjunto de OID's) diferente, o que faz cada agente ter uma implementação diferente.

Para auxiliar na implementação, foi utilizada a SNMP4J, que provê uma parte de sua API open source, e ajudou a acessar essas informações. Para o envio das tarefas para impressão foi utilizado o protocolo IPP (Internet Printer Protocol). O próprio Java já implementa as classes para mandar tarefas de impressão através desse protocolo, as únicas coisas necessárias para enviar é a URI da impressora e o formato do arquivo (através da classe DocFlavor é possível definir o formato do arquivo e posteriormente a API do Java se encarrega de converter para um formato que a impressora entende).

A comunicação com agente tradutor ocorre sempre quando o agente impressora deseja trocar informações com os outros agentes do sistema ou quando é notificada de alguma mensagem em sua área. Dessa forma, o agente impressora também precisa implementar os protocolos de comunicação com o agente tradutor. Além disso, como validará queries em SPARQL vindas do agente servidor, é necessário ter um módulo de reasoning implementado.

### 5.1.5 Agente Servidor

O agente servidor verifica e cadastra permissões de acesso de um usuário nas impressoras, escalona as tarefas de impressão e agrega impressoras em grupos. A verificação de permissões é relizada quando o agente servidor recebe uma solicitação de enviar um trabalho para a impressora.

Como o agente usuário já deve estar cadastrado no agente servidor, o tipo desse usuário já é conhecido. A relação entre as permissões e as impressoras é representada através de uma ontologia, e as solicitações de impressão serão somente enviadas para as impressoras que podem prestar serviço para determinado usuário.

A representação da ontologia de permissão de acesso às impressoras não faz parte

dos módulos de ontologia descritos anteriormente, pois se trata de uma funcionalidade específica do agente servidor e não é uma informação que seja utilizada para alinhamento de conceito entre os dispositivos.

Tabela 5.8: Ontologia de permissão

| *Classes* | *Propriedades de objetos* |
|---|---|
| ImpressoraGenerica | hasPermission |
| TipoUsuario | hasSubclass |
| Usuario | hasUser |

Fonte: O Autor

A tabela 5.8 mostra a ontologia de permissão, que é bastante simples. Utiliza os três conceitos : *ImpressoraGenerica*, *Usuario* e *TipoUsuario*. A *ImpressoraGenerica* é um conceito das impressoras que estão conectadas com o agente servidor. O *TipoUsuario* é o conceito que ilustra um tipo de usuário, ele pode ser representado como uma árvore através da propriedade de dado *hasSubclass* e se relaciona com o conceito de usuário (Usuario) com a propriedade de dado *hasUser*. Além disso, tem uma propriedade *hasPermission* que tem como *domain* a classe *TipoUsuario* e como *range* a classe ImpressoraGenerica, ou seja relaciona um grupo de usuários a uma impressora.

As atribuições do usuário ao tipo de usuário e do tipo de usuário às impressoras se dão manualmente no agente servidor. Além disso, é papel do servidor descobrir impressoras no ambiente. Para isso, é necessário o servidor criar uma consulta SPARQL para buscar as impressoras. A consulta é enviada sempre para todos que são cadastrados no agente tradutor. É demostrado um modelo de consulta SPARQL feita pelo agente servidor para a descoberta de impressoras abaixo:

*SELECT ?id*

*WHERE* {

*?impressora http://inf.ufrgs.br/servidor#modeloImpressora ?id*

}

Outro papel do agente servidor é controlar quando uma impressora não tem mais tinta ou papel e fazer com que o agente usuário evite essas impressoras até o problema estar resolvido. Para isso, o agente servidor deve ser constantemente informado dos dados de nível de tinta e quantidade de papel das impressoras. Como descrito anteriormente, o agente impressora tem o papel de obter essas informações, então o papel do agente servidor é solicitá-las.

### 5.1.6 Agente Usuário

Para os agentes do tipo usuário, a implementação foi divida em duas partes, a primeira é a interação com o usuário, que envolve as tarefas de capturar a impressão do usuário, e a segunda parte é a interação com o agente tradutor que envolve implementar os protocolos de comunicação.

Na primeira etapa foi criada uma fila de impressão virtual e o agente usuário verifica esporadicamente essa fila. Quando o usuário está em uma aplicação e solicita a impressão de algum documento, o documento e os parâmetros de impressão viram uma tarefa de impressão e são direcionados para a fila de impressão. Uma vez na fila de impressão, o agente usuário percebe a presença de uma nova tarefa e retira da fila de impressão. A tarefa passa por um processo de adaptação semântica, onde os parâmetros de impressão e informações adicionais sobre a tarefa são **anotados** semanticamente.

O agente usuário interage com o agente tradutor em algumas situações:

- Em sua criação, a primeira interação com o agente tradução é o seu cadastro.
- Após o cadastro no agente tradutor, o agente usuário deve procurar um agente servidor para gerenciar suas permissões e distribuição de tarefas, essa procura é feita através do agente tradutor, postando mensagem para todos os agentes cadastrados.
- A outra situação é quando o agente usuário precisa enviar uma mensagem para algum agente.

Para satisfazer a primeira situação basta implementar o protocolo tratado na figura 4.3. Após ser cadastrado no agente tradutor, o agente usuário já tem uma área correspondente para receber mensagem e a segunda situação pode acontecer. O agente usuário utiliza o protocolo de envio de mensagem (mostrado na figura 4.4) para o agente tradutor buscando um agente servidor. A pesquisa é feita criando uma consulta em SPARQL, considerando o vocabulário de sua própria ontologia para buscar o conceito desejado, conforme a consulta abaixo:

*SELECT ?impressora*
*WHERE* {
*?impressora rdf:type http://inf.ufrgs.br/usuario/joao#Impressora*
}

O retorno da mensagem é realizada pelo agente servidor, postando o seu identificador de área no agente tradutor, em resposta à mensagem do agente usuário. Dessa

forma, o agente usuário mantém o número do seu servidor para posterior comunicação.

Quando um agente usuário quiser enviar uma mensagem para algum agente, utiliza o mesmo protocolo de postar uma mensagem da situação anterior. Como foi visto no subcapítulo "Descrição dos conjuntos de ontologia", todos os serviços do usuário estão descritos na ontologia do agente através de pré-Condições, pós-condições e etapas. Para um serviço ser prestado todas as pré-condições devem ter sido atendidas. Dessa forma o agente usuário pode verificar qual ação irá executar. O serviço pode ser executado em etapas, mas para isso é necessário que o agente grave o estado de cada serviço que está sendo prestado. Cada uma tem associado um protocolo que é uma mensagem enviada e a resposta esperada. Após o serviço ser executado, como resultado tem a pós-condição.

## 5.2 Resultados

Esse capítulo apresenta os resultados da implementação do estudo de caso aplicando os agentes propostos nesse trabalho. O algoritmo de similaridade utilizado para o alinhamento será o primeiro a ser analisado nesse capítulo. Existem alguns elementos das tabelas que não possuem mapeamento e nem valor de similaridade associados, isso porquê todos os mapeamentos realizados tiveram similaridade menor que 0.3, que foi considerado o ponto de corte para o algoritmo. Nas tabelas 5.10, 5.9 e 5.11 representam o mapeamento realizado entre as classes dos padrões propostos.

Tabela 5.9: Similaridade entre classes padrão HP e padrão Epson

| *Classe HP* | *Classe Epson* | *Similaridade* |
|---|---|---|
| HPImpressora | EpsonImpressora | 0.615384 |
| HPCartucho | - | - |
| HPProtocolo | EpsonProtocolo | 0.583333 |
| HPMensagem | EpsonMensagem | 0.545454 |
| HPMedidas | EpsonMedidas | 0.500000 |
| HPServicos | EpsonServicos | 0.545454 |

Fonte: O Autor

Tabela 5.10: Similaridade entre classes padrão Epson e padrão Lexmark

| *Classe Epson* | *Classe Lexmark* | *Similaridade* |
|---|---|---|
| EpsonImpressora | ImpressoraLexmark | 0.533333 |
| EpsonProtocolo | ProtocoloLexmark | 0.500000 |
| EpsonMensagem | MensagemLexmark | 0.461538 |
| EpsonMedidas | MedidasLexmark | 0.416666 |
| EpsonServicos | ServicosLexmark | 0.461538 |

Fonte: O Autor

Tabela 5.11: Similaridade entre classes padrão Lexmark e padrão HP

| *Classe Lexmark* | *Classe HP* | *Similaridade* |
|---|---|---|
| ImpressoraLexmark | HPImpressora | 0.533333 |
| ProtocoloLexmark | HPProtocolo | 0.500000 |
| MensagemLexmark | HPMensagem | 0.461538 |
| MedidasLexmark | HPMedidas | 0.416666 |
| ServicoLexmark | HPServicos | 0.461538 |

Fonte: O Autor

Na tabela 5.9 e na tabela 5.11, que apresentam o mapeamento entre o padrão HP com os outros dois padrões (Epson e Lexmark) a classe HPCartucho não aparece relacionada a nenhum conceito dos padrões Epson e Lexmark, justamente por não ter nenhum correspondente nesses padrões. As tabelas 5.12, 5.13 e 5.14 apresentam o resultado do algoritmo de similaridade para as propriedades de dados dos padrões das impressoras.

Tabela 5.12: Similaridade entre propriedades de dados padrão HP e padrão Epson

| *Propriedade de dado HP* | *Propriedade de dado Epson* | *Similaridade* |
|---|---|---|
| modeloImpressora | modeloImpressora | 1.000000 |
| descricaoImpressora | descricao_impressora | 0.833333 |
| tipoDuplex | tipo_duplex | 0.666666 |
| memoriaRAM | memoria_ram | 0.666666 |
| capacidadePapel | capacidade_papel | 0.785714 |
| modeloCartucho | - | - |
| tipoImpressao | - | - |
| tipoTinta | tipo_tinta | 0.625000 |
| nomeProtocolo | nome_protocolo | 0.750000 |
| descricaoProtocolo | descricao_protocolo | 0.823529 |
| performative | performative | 1.000000 |
| conteudo | content | 0.500000 |
| receiver | receiver | 1.000000 |
| step | step | 1.000000 |
| nomeMedida | nome_medida | 0.666666 |
| valorMedida | valor_medida | 0.700000 |
| nomeServico | nome_servico | 0.700000 |
| descricaoServico | descricao_servico | 0.800000 |

Fonte: O Autor

Na tabela 5.12, é possível destacar duas propriedades de dados. A primeira é o modeloCartucho do padrão HP, que não há nenhuma outra propriedade correspondente e todas as propriedades que foram testadas a similaridade ficaram abaixo do ponto de corte. A outra propriedade é a tipoImpressao, também do padrão HP, que foi mapeada para a propriedade modeloImpressora. Esse mapeamento incorreto levará a uma inconsistência na ontologia de alinhamento que deverá ser corrigida por intervenção humana.

Por terem o mesmo nome, os alinhamentos entre as propriedades de objetos são

Tabela 5.13: Similaridade entre propriedades de dado padrão Epson e padrão Lexmark

| *Propriedade de dado Epson* | *Propriedade de dado Lexmark* | *Similaridade* |
|---|---|---|
| modeloImpressora | modeloimpressora | 1.000000 |
| descricao_impressora | descricaoimpressora | 0.833333 |
| tipo_duplex | tipoduplex | 0.666666 |
| memoria_ram | memoriaram | 0.666666 |
| capacidade_papel | capacidadepapel | 0.785714 |
| tipo_tinta | tipotinta | 0.625000 |
| nome_protocolo | nome_protocolo | 1.000000 |
| descricao_protocolo | descricao_protocolo | 1.000000 |
| performative | performativa | 0.900000 |
| content | conteudo | 0.500000 |
| receiver | receiver | 1.000000 |
| step | step | 1.000000 |
| nome_medida | nomemedida | 0.666666 |
| valor_medida | valormedida | 0.700000 |
| nome_servico | nomeservico | 0.700000 |
| descricao_servico | descricaoservico | 0.800000 |

Fonte: O Autor

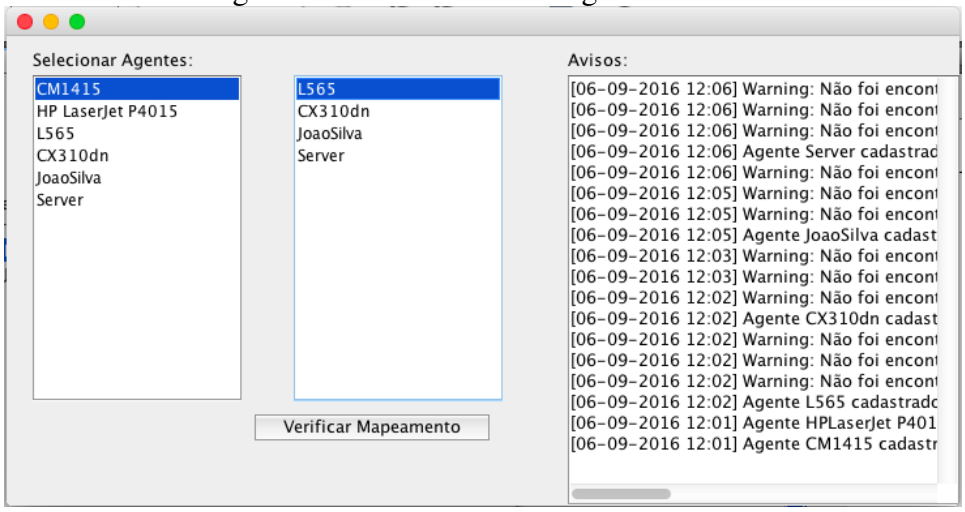
triviais. O que resta analisar são as interfaces dos agentes propostos. Inicialmente, o agente de alinhamento não tem uma interface para usuário, é rodado em background para auxiliar o agente tradutor. A interface do agente tradutor está apresentada nas figuras 5.4 e 5.5.

A primeira tela, figura 5.4, demonstra a imagem inicial do agente tradutor. O usuário pode selecionar os agentes para verificar o mapeamento realizado pelo sistema (figura 5.5, além disso, é possível observar na parte direita da tela um log de avisos onde aparecem todas as informações importantes do agente tradutor.

Assim como o agente de alinhamento, o agente usuário não tem interface principal para o usuário, roda em background na máquina do usuário verificando periodicamente a fila de impressão do usuário. As únicas interfaces que o agente usuário tem é a figura 5.6 que é o login para o agente acessar seu usuário e a figura 5.7 que é um aviso de qual impressora foi selecionada para impressão após a solicitação de impressão pelo usuário.

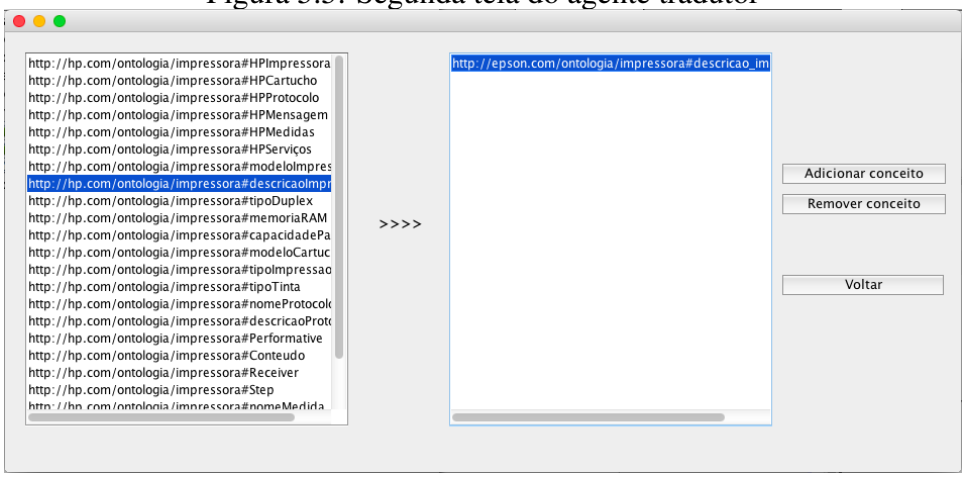
O agente servidor tem duas principais interfaces, uma para acompanhar o estado das impresssoras que foram descobertas e os usuários cadastrados no servidor (figura 5.8), e outra interface para ajuste de permissão de usuário para os grupos (figura 5.9). O agente impressora não tem nenhuma interface gráfica disponível ao usuário.

Figura 5.4: Primeira tela do agente tradutor



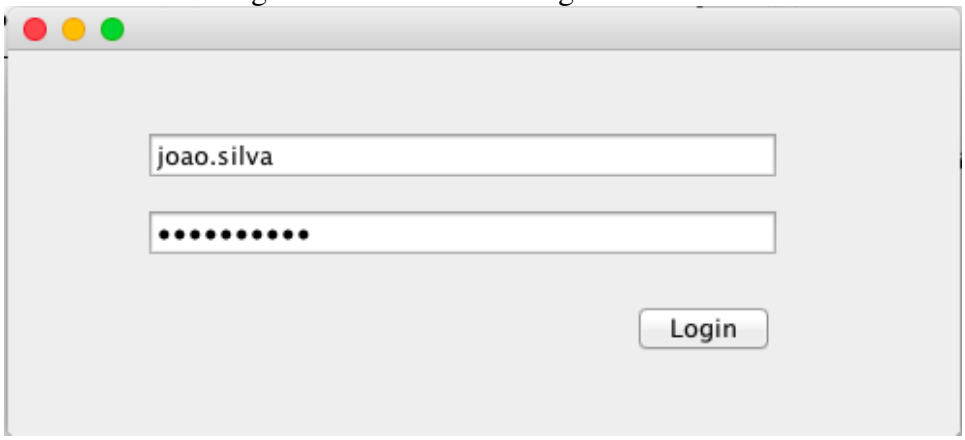

Fonte: O Autor

Figura 5.5: Segunda tela do agente tradutor



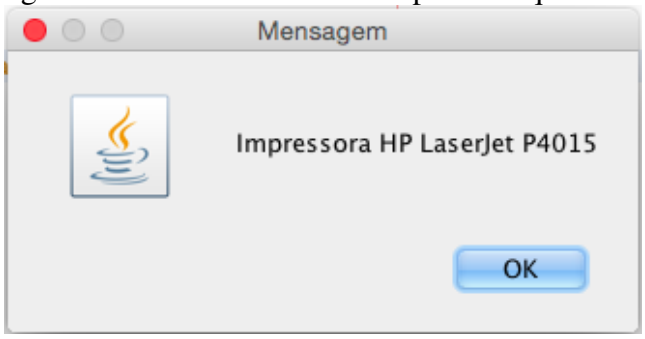

Fonte: O Autor

Figura 5.6: Interface de login do usuário



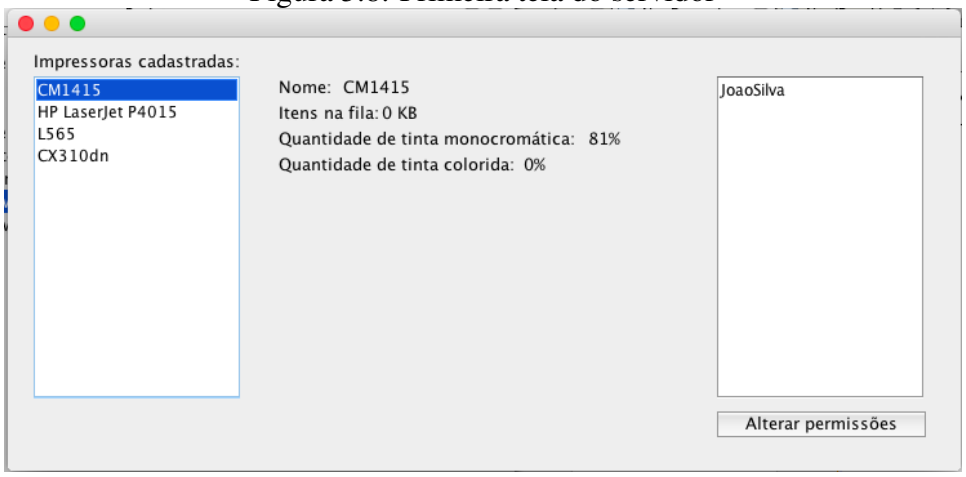

Fonte: O Autor

Figura 5.7: Agente usuário informando a impressora que executou a tarefa



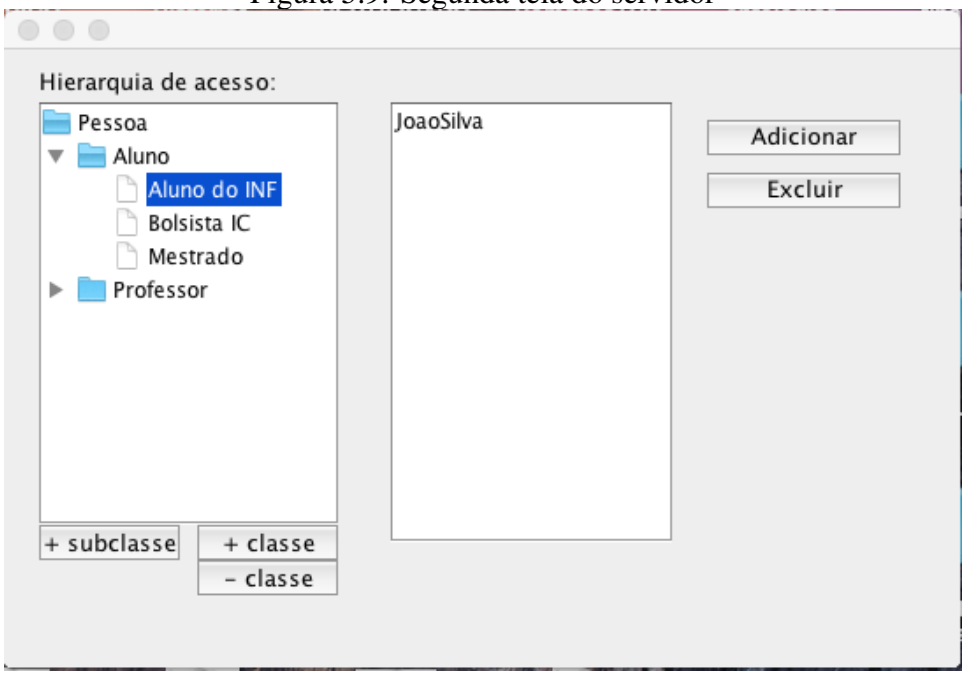

Fonte: O Autor

Figura 5.8: Primeira tela do servidor




Fonte: O Autor

Figura 5.9: Segunda tela do servidor




Fonte: O Autor

Tabela 5.14: Similaridade entre propriedades de dado padrão Lexmark e padrão HP

| *Propriedade de dado Lexmark* | *Propriedade de dado HP* | *Similaridade* |
|---|---|---|
| modeloImpressora | modeloimpressora | 1.000000 |
| descricaoimpressora | descricaoImpressora | 1.000000 |
| tipoduplex | tipoDuplex | 1.000000 |
| memoriaram | memoriaRAM | 1.000000 |
| capacidadepapel | capacidadePapel | 1.000000 |
| tipotinta | tipoTinta | 1.000000 |
| nome_protocolo | nomeProtocolo | 0.750000 |
| descricao_protocolo | descricaoProtocolo | 0.823529 |
| performativa | performative | 0.900000 |
| conteudo | conteudo | 1.000000 |
| receiver | receiver | 1.000000 |
| step | step | 1.000000 |
| nomemedida | nomeMedida | 1.000000 |
| valormedida | valorMedida | 1.000000 |
| nomeservico | nomeServico | 1.000000 |
| descricaoservico | descricaoServico | 1.000000 |

Fonte: O Autor

# 6 CONCLUSÃO E TRABALHOS FUTUROS

A ideia de implementar um agente, utilizando a arquitetura blackboard para tentar resolver o problema de interoperabilidade semântica parece ser promissora. Mesmo com um algoritmo de similaridade simples, como é o caso do N-GRAM, o alinhamento demonstrou um bom resultado. Entretanto alguns pontos ainda precisam ser melhorados:

- *Flexibilizar acesso ao agente tradutor*: O acesso ao agente que contém o blackboard não é flexível, ou seja depende de um protocolo bastante específico para cadastro, postagem e leitura das mensagens. Deve ser pensado como trabalho futuro uma solução para tratar esse problema.
- *Desempenho do algoritmo de alinhamento*: Apesar de separar o agente tradutor do agente de alinhamento, o desempenho do algoritmo de alinhamento ainda é um problema. Nos experimentos realizados com ontologias de tamanho médio, cada mapeamento demorou por volta de um minuto.
- *Precisão do algoritmo de alinhamento*: O algoritmo escolhido apenas considerava os termos das ontologias, sem considerar a estrutura ou a semântica dos conceitos. Como trabalho futuro seria interessante acrescentar um thesaurus, que buscará termos relacionados, de acordo com o contexto, com a palavra que será analisada, dessa forma a análise do termo seria mais ampla. Além disso, utilizar outros parâmetros para avaliar a similaridade, como por exemplo a estrutura ou uma análise semântica.
- *Criar ontologia para o agente tradutor*: Criar uma ontologia para o agente tradutor visando a interoperabilidade entre o agente tradutor e os agentes que estão cadastrados nele. Além disso, pode tornar mais compreensíveis os metadados das mensagens postadas.
- *Necessidade de SPARQL*: Os agentes participantes do agente tradutor precisam suportar consultas SPARQL. Essa limitação pode ser suprimida com a criação de um agente específico que realize esse serviço no sistema.

# REFERÊNCIAS

ABBURU, S. A survey on ontology reasoners and comparison. **International Journal of Computer Applications**, Foundation of Computer Science, v. 57, n. 17, 2012.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. **Comput. Netw.**, v. 54, 2010.

AUSTIN, J. L. **How to do things with words**. [S.l.]: Oxford university press, 1975.

BAKER, A. Complete manufacturing control using a contract net: a simulation study. In: IEEE. **Computer Integrated Manufacturing, 1988., International Conference on**. [S.l.], 1988. p. 100–109.

BAUMGARTNER, P.; FURBACH, U.; NIEMELÄ, I. Hyper tableaux. In: SPRINGER. **European Workshop on Logics in Artificial Intelligence**. [S.l.], 1996. p. 1–17.

CASTELLANI, A. P. et al. Web services for the internet of things through coap and exi. In: IEEE. **Communications Workshops (ICC), 2011 IEEE International Conference on**. [S.l.], 2011. p. 1–6.

COLLINA, M.; CORAZZA, G. E.; VANELLI-CORALLI, A. Introducing the qest broker: Scaling the iot by bridging mqtt and rest. In: IEEE. **Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on**. [S.l.], 2012. p. 36–41.

DEMAZEAU, Y. From interactions to collective behaviour in agent-based systems. In: CITESEER. **In: Proceedings of the 1st. European Conference on Cognitive Science. Saint-Malo**. [S.l.], 1995.

DENTLER, K. et al. Comparison of reasoners for large ontologies in the owl 2 el profile. **Semant. web**, 2011.

EUZENAT, J. Introduction to ontology matching and alignment. Citeseer, 2009.

HAYES-ROTH, B. A blackboard architecture for control. **Artif. Intell.**, 1985.

KHRIYENKO, O.; TERZIYAN, V.; KAIKOVA, O. User-assisted semantic interoperability in internet of things. In: **The Sixth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM'12)**. [S.l.: s.n.], 2012. p. 104–110.

KRAFZIG, D.; BANKE, K.; SLAMA, D. **Enterprise SOA: Service-Oriented Architecture Best Practices (The Coad Series)**. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2004. ISBN 0131465759.

MORAIS, E. A. M.; AMBRÓSIO, A. P. L. Ontologias: conceitos, usos, tipos, metodologias, ferramentas e linguagens. **Relatório Técnico–RT-INF-001/07, dez**, 2007.

NIKRAZ GIOVANNI CAIRE, P. A. B. M. A methodology for the analysis and design of multi-agent systems using jade. **International Journal of Computer Systems Science and Engineering**, 2006.

OLIVEIRA, F. M. Inteligência artificial distribuída. In: SOCIEDADE BRASILEIRA DE COMPUTAçãO. **IV Escola Reginal de Informática**. [S.l.], 1996.

OWL-S: Semantic Markup for Web Services. <http://www.w3.org/Submission/OWL-S/>, note="Acessado em 2016-05-12".

RITCHIE, S. G. A knowledge-based decision support architecture for advanced traffic management. **Transportation Research Part A: General**, Elsevier, v. 24, n. 1, p. 27–37, 1990.

SERRANO, M. et al. Iot semantic interoperability: Research challenges, best practices, recommendations and next steps. **White Paper of the 4th Activity Chain of the EUROPEAN RESEARCH CLUSTER ON THE INTERNET OF THINGS (IERC)**, 2015.

SILVA, L. R. J. da; GLUZ, J. C. Mssearch: Busca semântica de objetos de aprendizagem obaa com suporte a alinhamento automático de ontologias. In: . [S.l.: s.n.].

SMITH, R. G. The contract net protocol: High-level communication and control in a distributed problem solver. **IEEE Transactions on Computers**, 2006.

STEINER, D. D. Imagine: an integrated environment for constructing distributed artificial intelligence systems. **Foundations of distributed artificial intelligence**, John Wiley & Sons, v. 9, p. 345, 1996.

SUTARIA, R.; GOVINDACHARI, R. Making sense of interoperability: Protocols and standardization initiatives in iot. In: . [S.l.: s.n.], 2013.

VEER, H. van der; WILES, A. Achieving technical interoperability. **European Telecommunications Standards Institute**, 2008.

WANG, W. et al. A comprehensive ontology for knowledge representation in the internet of things. In: IEEE. **Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on**. [S.l.], 2012. p. 1793–1798.

WEISS, G. **Multiagent systems: a modern approach to distributed artificial intelligence**. [S.l.]: MIT press, 1999.

XU, H.; BRUSSEL, H. V. A behaviour-based blackboard architecture for reactive and efficient task execution of an autonomous robot. **Robotics and Autonomous Systems**, Elsevier, v. 22, n. 2, p. 115–132, 1997.

ZHANG, B. et al. Agent architecture: a survey on robocup-99 simulator teams. In: IEEE. **Intelligent Control and Automation, 2000. Proceedings of the 3rd World Congress on**. [S.l.], 2000. v. 1, p. 194–198.