UNIVERSIDADE ESTADUAL PAULISTA
"Júlio de Mesquita Filho"
Pós-Graduação em Ciência da Computação

Renato Luciano Cagnin

Uma Arquitetura Multiagente para gerenciamento de dispositivos em Ambientes da Internet das Coisas

UNESP/ Rio Claro
2015

Cagnin, Renato Luciano.
Uma arquitetura multiagente para gerenciamento de dispositivos em ambientes da internet das coisas / Renato Luciano Cagnin. -- São José do Rio Preto, 2015
123 f. : il., tabs.

Orientador: Ivan Rizzo Guilherme
Dissertação (mestrado) – Universidade Estadual Paulista "Júlio de Mesquita Filho", Instituto de Biociências, Letras e Ciências Exatas

1. Computação - Matemática. 2. Internet das coisas. 3. Sistema multiagente. 4. Arquitetura orientada a serviços (Computação) 5. Web semântica. 6. Ontologias (Recuperação da informação) I. Guilherme, Ivan Rizzo. II. Universidade Estadual Paulista "Júlio de Mesquita Filho". Instituto de Biociências, Letras e Ciências Exatas. III. Título.
CDU – 518.72

Ficha catalográfica elaborada pela Biblioteca do IBILCE
UNESP - Câmpus de São José do Rio Preto

Renato Luciano Cagnin

Uma Arquitetura Multiagente para gerenciamento de dispositivos em Ambientes da Internet das Coisas

Orientador: Prof. Dr. Ivan Rizzo Guilherme

Dissertação de Mestrado elaborada junto ao Programa de PósGraduação em Ciência da Computação - Área de Concentração em Sistemas de Computação, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação

UNESP/Rio Claro
2015

Renato Luciano Cagnin

Uma Arquitetura Multiagente para gerenciamento de dispositivos em Ambientes da Internet das Coisas

Dissertação de Mestrado elaborada junto ao Programa de Pós-Graduação em Ciência da Computação – Área de Concentração em Sistemas Inteligentes, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

BANCA EXAMINADORA

Prof. Dr. Ivan Rizzo Guilherme

UNESP – Universidade Estadual Paulista Júlio de Mesquita Filho - Rio Claro
Orientador

Prof. Dr. Paulo André Lima De Castro

Instituto Tecnológico de Aeronáutica (ITA)

Profa. Dra. Simone das Graças Domingues Prado

UNESP – Universidade Estadual Paulista Júlio de Mesquita Filho - Bauru

UNESP
Rio Claro - 2015

AGRADECIMENTOS

Agradeço primeiramente à minha família por tudo que fizeram ao longo da minha caminhada nesta vida.

À minha amada Andréa pelo carinho, amor e compreensão, mesmo em momentos de distância e ausência.

Ao Prof. Dr. Ivan Rizzo Guilherme pela orientação deste trabalho e experiências passadas.

Aos amigos Jonas e Rodrigo pelo grande incentivo e contribuição no desenvolvimento deste trabalho.

Ao meu grande amigo e praticamente irmão Vinícius Soares por sua amizade e companheirismo ao longo de todos estes anos.

Ao meu grande amigo Tommy por todos estes anos de amizade. Aos amigos Alan, William, Heleno e Ferri.

Aos grandes amigos Julio César e Cínthia pelos bons momentos que passamos.

Aos amigos do curso de Ciência da Computação.

Aos demais professores et al. que me auxiliaram durante minha graduação e pós-graduação, meus mais sinceros agradecimentos.

RESUMO

A Internet das Coisas (IoT) é o termo usado para definir o novo cenário da Internet, caracterizado pela conexão de uma grande variedade de dispositivos que possuem interfaces de comunicação sem fio e estão imersos em um ambiente físico. Na IoT, esses ambientes são abertos, onde novos componentes podem ser incorporados; distribuídos, caracterizados por diversos componentes conectados em rede; e dinâmicos. Tais características trazem diversos desafios para o desenvolvimento de aplicações que buscam acessar, integrar e analisar a quantidade de dados produzida por estes dispositivos. Nesse sentido, este trabalho propõe uma arquitetura de software para o desenvolvimento de aplicações no contexto da IoT. A arquitetura proposta é constituída de diversas camadas, caracterizadas por diferentes tecnologias. As tecnologias utilizadas são Sistemas Multiagente, Arquitetura Orientada a Serviços e Web Semântica. Para demonstrar a viabilidade da proposta, um protótipo da arquitetura foi desenvolvido e aplicado em um estudo de caso no contexto da automação residencial. Os resultados apresentados demonstram que a arquitetura demonstrou capacidade de identificação de diferentes dispositivos, operações e serviços; e coordenação da execução de operações de dispositivos em tarefas de maior complexidade segundo informações contextuais.

Palavras - Chave: Internet das Coisas, Sistemas Multiagente, Arquitetura orientada a serviços,Tecnologias da Web Semântica

ABSTRACT

The Internet of Things (IoT) is the term used to define the new Internet scenario, characterized by connecting a variety of devices that have wireless communication interfaces and are immersed in a physical environment. The IoT environments are open, where new components can be incorporated; distributed, characterized by several components networked; and dynamic. These characteristics bring many challenges for the development of applications that seek to access, integrate and analyze the amount of data produced by these devices. In this sense, this paper proposes a software architecture for the development of applications in the context of IoT. The proposed architecture is made up of several layers characterized by different technologies. The technologies used are Multi-Agent Systems, Service Oriented Architecture and Web Semantics. To demonstrate the feasibility of the proposal, a prototype of the architecture was developed and applied in a case study in the context of home automation. The results presented show that the architecture demonstrated ability to identify different devices, operations, and services; and coordinate the executions of operation of devices in more complex tasks according to the available contextual information.

Keywords: Internet of Things, Multi-Agent Systems , *Service Oriented Architecture*, Web Semantic Technologies

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIAÇÔES

6LoWPan: *IPv6 over Low power Wireless Personal Area Networks*

API: *Application Programming Interface*

BDI: *Belief, Desire, Intention*

CArtAgO: *Common ARtifact infrastructure for AGent Open environment*

DARPA KSE: *DARPA Knowledge Sharing Effort*

DPWS: *Devices Profile for Web Services*

HTTP: *Hypertext Transfer Protocol*

IoT: *Internet of Things*

IP: *Internet Protocol*

JaCaMo: Jason, CArtAgO, Moise

JMEDS: Java *Multi Edition* DPWS *Stack*

KQML: *Knowledge Query and Manipulation Language*

LAN: *Local Area Network*

Moise: *Model of Organisation for multI-agent SystEms*

OML: *Organisation Modelling Language*

OPC-UA: *OPC Unified Architecture*

ORA4MAS: *Organisation Artifacts for Multi-agent systems*

OWL: *Web Ontology Language*

RDF: *Resource Description Framework*

REST: *Representational State Transfer*

SMA: Sistemas Multiagentes

SOA: *Service-Oriented Architecture*

SOAP: *Simple Object Access Protocol*

SWOT: *Semantic Web of Things*

UDDI: *Universal Description, Discovery and Integration*

UML: *Unified Modeling Language*

XML: *eXtensible Markup Language*

W3C: *World Wide Web Consortium*

WSDL: *Web Services Description Language*

SUMÁRIO

1 INTRODUÇÃO

A Internet das Coisas (*Internet of Things* - IoT) provê uma infraestrutura de rede de dispositivos heterogêneos, comunicando uns com os outros em um ambiente aberto e dinâmico. Segundo Guillemin e Friess (2009), a IoT permite que as pessoas possam se conectar com qualquer coisa (dispositivos) quando e onde quiserem, usando qualquer caminho, rede ou serviço. Isso significa que a IoT promete criar um mundo onde todos os dispositivos presentes nos diversos ambientes estarão conectados à Internet. Estes dispositivos serão capazes de interagir uns com os outros, publicar e dar acesso aos seus recursos e realizar tarefas de maneira autônoma, com o mínimo de intervenção humana (PERERA et al., 2014).

A Internet das coisas é um campo de pesquisa que surgiu com o aumento do uso de dispositivos em diversos domínios. Atualmente, existem milhões de dispositivos conectados a internet em todo o mundo, e este número cresce a cada dia. A previsão é de 50 a 100 bilhões de dispositivos conectados à Internet até 2020 (SUNDMAEKER et al., 2010). Esses dispositivos conectados na Internet coletam e fornecem dados de seu ambiente para outros sistemas e aplicações.

Este aumento se deve aos avanços em *hardware* e infraestrutura de rede que têm contribuído cada vez mais para o desenvolvimento de dispositivos mais baratos, menores, dotados de mais recursos de processamento, e também interfaces de comunicação sem fio. Segundo Wang (2004), a principal característica desses dispositivos é a combinação do elemento sensor, processamento de informação e tecnologia de comunicação, que os permite funcionalidades mais avançadas do que simplesmente disponibilizar dados brutos.

Dispositivos com tais características têm sido usados em uma variedade de ambientes, tais como casa, indústria e áreas urbanas e rurais (RUTA et al., 2013). O desenvolvimento de sistemas que fazem uso dos dados e ações fornecidos por estes dispositivos de maneira autônoma denomina-se Inteligência Ambiental.

A visão da Inteligência Ambiental se refere a um ambiente, apresentando diversos dispositivos heterogêneos, que podem disponibilizar dados e realizar ações para os usuários de maneira autônoma de acordo com a necessidade de usuários (STAVROPOULOS et al., 2013). Dessa forma, estes ambientes apresentam sistemas de monitoramento e controle com a capacidade de monitorar e alterar o ambiente físico, segundo a análise comportamental, ou requisições realizadas pelos usuários. Neste caso, interfaces específicas são necessárias para facilitar o controle tanto dos dispositivos quanto das características do ambiente.

Estes sistemas de monitoramento de ambientes são desenvolvidos considerando fornecer aos usuários informações sobre as condições do ambiente ou das entidades presentes no mesmo, de maneira rápida, econômica e efetiva. Um exemplo de implementação no contexto de áreas urbanas é denominado iBus (BOYLE; YATES; YEATMAN, 2013), sistema que integra GPS (para dados de localidades), transceptor GPRS (para conectividade de IP móvel, portanto, a transferência de dados), e medições físicas (incluindo odômetros e giroscópios). A implantação do sistema em todos os 8.000 ônibus da cidade de Londres permite com que os usuários possam visualizar a velocidade, direção, e local de cada ônibus, a partir de dados transmitidos em tempo real.

Na maioria das vezes aplicações gerenciando a execução de dispositivos em ambientes da IoT têm como objetivo fornecer aos usuários informações para monitorar e controlar variáveis desse ambiente. Em geral estas aplicações buscam prover melhorias no conforto e na qualidade de vida humana (LEONG et al., 2014). Por exemplo, casas inteligentes são projetadas para proporcionar entretenimento multimídia e informação em todos os cômodos, onde dispositivos eletrônicos móveis e redes de sensores distribuídos são usados para o monitoramento e controle das condições (luz, som, temperatura, umidade, entre outros) e dos objetos presentes do ambiente; assim como para a interação com o usuário (MOELLER; SLEMAN, 2008). Contudo, tais aplicações apresentam diversos desafios que surgem devido à eficiência de energia, segurança e privacidade, múltiplos protocolos de conexão e complexidade dos ambientes (GUBBI et al.,2013).

No contexto de eficiência de energia, melhorias têm sido desenvolvidas a nível de *hardware, software* e protocolos de comunicação. Neste sentido, dispositivos tem apresentado baterias menores e mais duráveis; *softwares* em dispositivos embarcados e algoritmos que otimizam o consumo de energia; e protocolos de comunicação, como 6LoWPan (SAMARAS et al., 2013).

No contexto de segurança e privacidade, as soluções encontradas em redes de menor escala, como ambientes industriais e casas, consistem na adoção de redes *ad-hoc*. Na questão de privacidade de usuários, entretanto existem problemas ainda em aberto, pois os dispositivos ainda não são desenvolvidos para usuários específicos ou acessos restritos. Apesar destes problemas, existem tecnologias de interoperabilidade, como DPWS e OPC-UA, que adicionam protocolos de segurança permitindo que os desenvolvedores das aplicações possam utilizar dados criptografados durante o acesso e execução de operações (DIAS; REGIS,2012).

Além das questões de segurança e eficiência de energia, aplicações em ambientes da IoT devem garantir a interoperabilidade entre os dispositivos. Este problema surge a partir das diferentes tecnologias de comunicação e modelos de dados utilizados por esses diferentes componentes. Neste sentido, arquiteturas orientadas a serviços (SOA) consistem em uma abordagem promissora para apoiar a interoperabilidade entre estes componentes (RUTA et al., 2013). Nos últimos anos, tecnologias baseadas em SOA para integração de dispositivos têm sido adotadas para lidar com esses problemas, através de infraestruturas caracterizadas por interfaces de comunicação padronizadas e mecanismos para publicação e descoberta de novos dispositivos conectados à rede (KARNOUSKOS; TARIQ, 2009).

Contudo o alto tempo de acesso e o excesso de complexidade, envolvidos para a implementação e implantação de tecnologias, tais como SOAP, XML e HTTP, são demasiado elevados para o uso em dispositivos e outros sistemas embarcados (SAMARAS et al., 2013). Para viabilizar o uso de SOA em redes de dispositivos, tecnologias específicas têm sido desenvolvidas minimizando a complexidade inerente e incorporando serviços específicos para dispositivos. Exemplos de tais tecnologias são o DPWS (*Devices Profile for Web Services*) e OPC-UA (IZAGUIRRE; LOBOV; LASTRA, 2011). Nestes casos, os dispositivos passam a ser provedores de serviços, além de suportarem mecanismos de *Plug&Play* (publicação, descoberta e consumo automático de recursos).

Além das questões de interoperabilidade, ambientes da IoT são bastante complexos, e sua complexidade surge tanto pela distribuição e quantidade de componentes heterogêneos (os dispositivos e sistemas), bem como pelo grande volume de dados produzidos e a alta comunicação existente entre os seus diversos componentes. As soluções adotadas para este problema envolvem a aplicação de técnicas da Inteligência Artificial como Sistemas Multiagentes (SMA) e tecnologias da Web Semântica.

Tecnologias da Web Semântica, como ontologias, também têm sido adotadas para auxiliar na interoperabilidade de dispositivos e na integração de dados heterogêneos. Essas tecnologias são empregadas para realizar a anotação semântica dos modelos de dados e automação das tarefas dos sistemas, através da definição de vocabulários comuns para representar e descrever o conhecimento utilizado nas aplicações (ALOULOU et al. 2013). Além disso, o uso de ontologias auxilia na identificação de domínios específicos, permitindo aos dispositivos extrair dados de dispositivos e objetos implantados no ambiente e processá-los automaticamente de forma consistente a um contexto, a fim de melhor apoiar as atividades de um usuário e satisfazer suas necessidades (ABOWD et al., 1999).

Neste sentido, o uso de tecnologias da Web Semântica tem apresentado um grande avanço na integração de dados de dispositivos (PERERA et al., 2014). A fusão das tecnologias da Web Semântica e da Internet das Coisas é denominada Web Semântica das Coisas de (*Semantic Web Of Things* (SWOT)), e esta tem por objetivo a disponibilização de informações semânticas do ambiente permitindo uma comunicação mais eficiente entre usuários e dispositivos ou mesmo entre os próprios dispositivos (RUTA et al., 2013).

No entanto, apenas integrar e prover os usuários com dados medidos a partir do ambiente pode não ser suficiente para resolver suas necessidades. Isso significa que em um ambiente inteligente as aplicações devem monitorar e controlar os dispositivos, fornecendo informações de alto nível e recursos para apoiar os usuários na realização de tarefas complexas, permitindo automatizar a execução de processos relacionados a tarefas.

Para o projeto e desenvolvimento de aplicações para esses ambientes são necessárias novas abordagens, capazes de prover ferramentas e tecnologias para assegurar a cooperação e coordenação dos dispositivos conectados. Nesse contexto, a abordagem de sistemas multiagentes (SMA) fornece um conjunto de abstrações apropriadas para lidar com o projeto e desenvolvimento de sistemas heterogêneos, distribuídos, abertos, dinâmicos e complexos (WOOLDRIDGE; JENNINGS, 1995). Nessa abordagem um conjunto de agentes são concebidos como componentes que agem de maneira independente e autônoma e são responsáveis por criar organizações e cooperar na realização de um conjunto de tarefas com o objetivo de atender as necessidades dos usuários.

1.1 Objetivos

O objetivo deste trabalho é propor uma arquitetura multiagente que suporte o desenvolvimento de aplicações voltadas para o gerenciamento de dispositivos em Ambientes de IoT. Para isso é proposto a utilização das tecnologias de SOA, Web Semântica e Sistemas Multiagentes, a fim de permitir o desenvolvimento de aplicações que apresentem comportamentos autônomos, capazes de identificar e se adaptar aos diferentes contextos, a partir da gestão de diferentes dispositivos heterogêneos.

1.2 Organização do texto

Neste trabalho, no Capítulo 2 são apresentados os aspectos conceituais, as técnicas e as ferramentas utilizadas para o desenvolvimento da abordagem proposta. No Capítulo 3, é

apresentada uma revisão da literatura a respeito de IoT e as tecnologias relacionadas, através da qual foram identificados e analisados os aspectos funcionais e requisitos dos ambientes e das aplicações nesse contexto. No Capítulo 4, é proposto a arquitetura multiagente e seus componentes, dando atenção especial aos papéis dos agentes, a estrutura social e funcional. No Capítulo 5 são descritos os componentes da arquitetura proposta, os quais são utilizados como parte do estudo de caso que foi desenvolvido. No Capítulo 6 é apresentado um estudo de caso no contexto da internet das coisas, onde o protótipo de um sistema para a automação residencial foi implementando e os testes foram realizado em um ambiente simulado. Finalmente, no Capítulo 7 são feitas considerações sobre as contribuições do trabalho e colocadas sugestões para trabalhos futuros.

2 ASPECTOS CONCEITUAIS

Neste capítulo, são apresentados os principais conceitos e tecnologias que estão relacionados ao desenvolvimento da abordagem proposta neste trabalho. Inicialmente é apresentada a definição de Internet das Coisas, seguida das descrições das metodologias e das tecnologias que serão empregadas na concepção da arquitetura e também no desenvolvimento do estudo de caso.

2.1. Internet das Coisas

A Internet das Coisas pode ser definida como a rede mundial de dispositivos conectados à Internet, exclusivamente com base em protocolos de comunicação padronizados, e endereçáveis (MURAR; BRAD, 2014). Segundo Pujolle (2006) a Internet das Coisas é um paradigma que lida com uma arquitetura que permite a tais dispositivos a troca de informações através da Internet.

A Internet das Coisas pode ser dividida em três paradigmas: orientado a coisas (dispositivos), orientado a internet (*middleware*) e orientado à semântica (conhecimento). Dessa forma, aplicações desenvolvidas para este domínio, devem apresentar funções que envolvem estes três paradigmas (ATZORI; IERA; MORABITO, 2010).

As "coisas", representadas principalmente por dispositivos na grande maioria dotados de conexões sem fio, são participantes ativas provendo informações e participando de processos sociais nos quais estão inseridas, interagindo e comunicando entre si e com o meio ambiente, podendo receber dados sobre o ambiente, e reagir a este de forma autônoma. Dessa forma, os acontecimentos do mundo real ou as requisições de usuários podem influenciar os dispositivos por meio da execução de processos que desencadeiam ações ou serviços. (SUNDMAEKER et al., 2010).

No entanto, segundo Gubbi et al. (2013) para a IoT ser viável, os cenários de computação terão que ir além de cenários tradicionais e devem evoluir incorporando inteligência aos objetos e dispositivos existentes.

Em cenários envolvendo a IoT, os usuários interagem com dispositivos (por exemplo, sensores ou dispositivos móveis) que coletam informações úteis para os outros usuários, em contextos sociais; ou informações do ambiente: como sensores de temperatura, câmeras de

segurança, sensores de localização, estado de saúde, entre outros. Além disso, podem requisitar ações a partir de dispositivos, permitindo o controle do ambiente, exibindo informações sobre eventos importantes na TV, controlando iluminação, recebendo mensagens de alarme, recebendo mensagens de outros usuários, etc. (BENAZZOUZ et al., 2014). Por outro lado, os recursos disponíveis na IoT podem ser utilizados por sistemas inteligentes, que possuem funcionalidades para a execução de serviços de maneira autônoma e pró-ativa para monitorar os ambientes e adaptar os estados dos mesmos segundo as preferências do usuário, por exemplo, visando o conforto, a segurança ou a vigilância do estado de saúde ou emocional dos usuários (BOYLE; YATES; YEATMAN, 2013).

O desenvolvimento de soluções para a Internet das Coisas, envolve características que devem ser consideradas ao longo da concepção, desenvolvimento, implementação e avaliação de tecnologias e recursos computacionais de hardware/software. Segundo Sundmaeker et al. (2010), as principais características são:

- Inteligência: significa que a aplicação deve gerar conhecimento através da aquisição de dados e da aplicação de raciocínio sobre estes, transformando dados brutos em conhecimento.
- Arquitetura: a arquitetura da IoT deve permitir tratar os dispositivos baseados em eventos (por exemplo, sensor da porta) e os que produzem dados de forma contínua em intervalos de tempo regulares (por exemplo, sensores de temperatura).
- Complexidade: a IoT compreende um grande número de objetos (sensores e atuadores) que interagem de forma autônoma. Novos objetos vão começar a se comunicar e os já existentes desaparecerão.
- Tamanho: o número de dispositivos conectados na IoT cresce continuamente assim como suas interações. Dessa forma, a IoT precisa facilitar a interação entre esses dispositivos.
- Tempo: devido ao elevado número de dispositivos, a IoT deve lidar com bilhões de eventos paralelos e simultâneos. O processamento de dados em tempo real é essencial.
- Espaço: a localização geográfica com precisão de um dispositivo desempenha um papel significativo na determinação de contextos. As interações entre os componentes são altamente dependentes de suas localizações, seus arredores e presença de outras entidades (por exemplo, objetos e pessoas).
- “Tudo como serviço”: todos os recursos computacionais (hardware e software) são encapsulados na forma de serviços, permitindo a interação entre homem-máquina e

máquina-máquina. Apesar de apresentar-se como um modelo eficiente, escalável e fácil de usar, também exige uma quantidade significativa de infraestrutura para ser colocada em prática.

Soluções para tais características envolvem outros requisitos que vão além dos dispositivos como sensores, atuadores com *hardware* de comunicação incorporados. Padrões de interoperabilidade, como por exemplo arquitetura orientada a serviços, permitem a transmissão e criação de grandes quantidades de dados que devem ser armazenados, processados e apresentados de uma forma contínua, eficaz e fácil de interpretar.

Devido ao grande volume de dados produzidos, são necessárias ferramentas de armazenamento e computação para a análise de dados e a determinação de contextos. Além de ferramentas e técnicas para apresentação, visualização, interpretação e a integração dos dados, outras devem ser definidas para permitir que os dados sejam amplamente acessados em plataformas diferentes e possam estar disponíveis para diferentes aplicações (GUBBI et al., 2013). Portanto, a infraestrutura necessária para a Internet das Coisas exige:

- Percepção da situação dos usuários e dispositivos inseridos no ambiente;
- Arquiteturas de *software* e redes de comunicação para processar e transmitir a informação contextual para onde é relevante;
- Ferramentas inteligentes de análise que permitam especificar localidade, conectividade e contextos.

2.2 Sistemas Multiagentes

Os Sistemas Multiagentes (SMA) consistem em uma abordagem de desenvolvimento de *software* proposta para a solução de problemas de sistemas complexos e distribuídos. Os SMA fornecem um conjunto de abstrações e ferramentas apropriadas para a modelagem e a construção de sistemas distribuídos (conhecimento, controle e recursos distribuídos), heterogêneos (integração de recursos e sistemas distintos) e abertos (novos componentes podem ser incluídos ou retirados do sistema) (PYNADATH; TAMBE, 2003). Estes sistemas são construídos segundo um conjunto de agentes, que desempenham diferentes papéis em uma organização, onde cooperam para realização de tarefas para a satisfação de objetivos.

2.2.1 Agentes

Os agentes apresentam-se como uma nova abordagem para a modelagem e o desenvolvimento de *softwares*. A definição de um agente é dada como um sistema computacional encapsulado situado em um determinado ambiente e dotado de autonomia para realização de determinados objetivos (RUSSELL; NORVIG, 2004).

Segundo Wooldridge e Jennings (1995), as principais características relacionadas a agentes são: autonomia, pró-atividade, reatividade e a habilidade social. Autonomia significa que um agente opera de maneira independente (sem a intervenção externa de um humano ou outros agentes), de modo a executar ações para realização de objetivos. Pró-atividade significa que o agente exibe um comportamento orientado a satisfação de um objetivo, sem que necessariamente seja invocado por entidades externas. Reatividade significa que o agente reage a mudanças em seu ambiente. E por fim, habilidade social é definida como a habilidade de um agente cooperar e coordenar suas ações com outros agentes (BORDINI; HÜBNER; WOOLDRIGE, 2007).

Além das características mencionadas acima, o agente também pode determinar qual é a forma de cumprir uma dada tarefa, que geralmente em sistemas baseados em agentes, é executada a partir de planos. Para a realização de tarefas, um agente, situado em um ambiente que pode ser físico (real) ou *software* (virtual), percebe este ambiente a partir de sensores e realiza ações a partir de atuadores.

O ambiente representa tudo que é externo ao agente. Para que um agente realmente esteja situado em um ambiente, pelo menos parte desse ambiente tem que ser observável e modificável pelo agente (MCARTHUR et al., 2007).

O ambiente também pode ser um ambiente comum compartilhado entre um conjunto de diferentes agentes. Caso os agentes realizem a interação entre si, tem-se o conceito de sistema multiagente (SMA). Estes agentes de um SMA têm autonomia para cooperar, coordenar e negociar, compartilhando conhecimento e executando as ações.

Nesses sistemas, os agentes exibem um comportamento autônomo e ao mesmo tempo interagem ou trabalham em conjunto em vista da satisfação de um objetivo próprio ou mesmo um objetivo que atenda ao sistema em geral (BRAZIER et al., 2009).

Portanto, estes agentes, que compõem um SMA, devem apresentar as seguintes características:

- Cooperação – capacidade de trabalharem juntos para alcançarem objetivos comuns.
- Coordenação – capacidade de gerir as interdependências entre as atividades, por exemplo, a utilização de recursos não compartilháveis.
- Negociação – capacidade de chegar a acordos sobre assuntos de interesse comum.

Devido às características citadas acima, em um SMA os agentes podem formar organizações que gerenciam o comportamento coletivo ou mesmo o comportamento de outras organizações. Nesse contexto, uma organização é utilizada para descrever e gerenciar o comportamento dos agentes em um dado cenário para a realização de uma ou mais tarefas. Também é papel da organização estabelecer restrições que devem ser respeitadas pelos agentes na realização das tarefas.

2.2.2 Programação Multiagente

A programação para sistemas multiagentes de acordo com Bordini et al. (2007), compreende especificar, como apresentado na Figura 1, três diferentes paradigmas: programação orientada a agentes, programação orientada ao ambiente e programação orientada à organização.

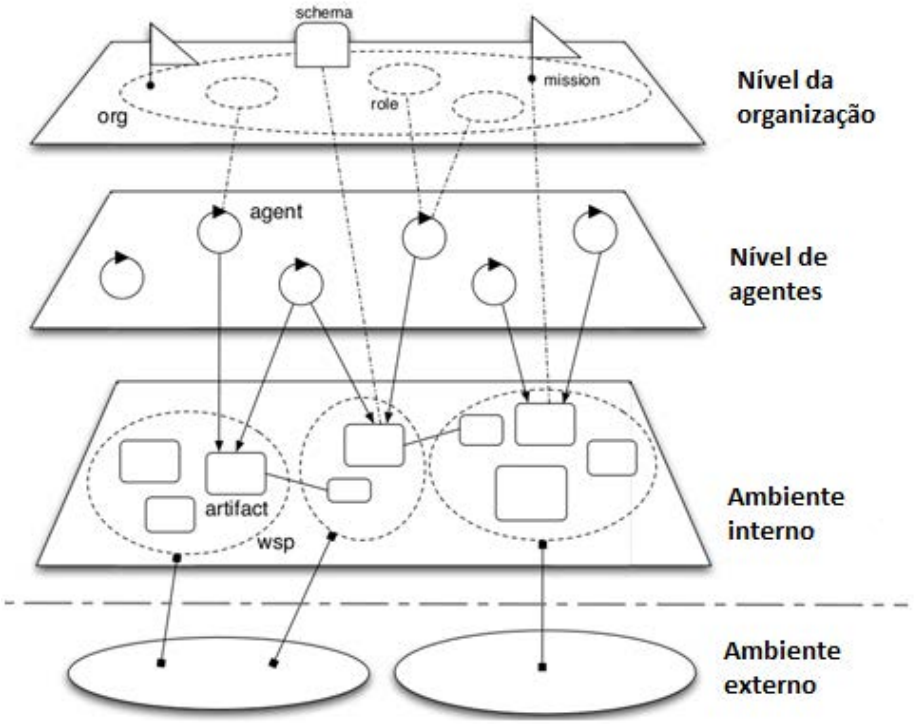

Figura 1: Arquitetura de sistemas multiagentes.
Adaptado de Bordini et al. (2007).

Na programação orientada a agentes, o programador desenvolve a camada do SMA responsável pelas ações/tarefas de cada agente. Neste caso, cada agente é considerado como uma entidade autônoma que interage com o ambiente. O ambiente do agente pode ser

desenvolvido a partir da programação orientada ao ambiente, onde são desenvolvidos os elementos sobre os quais os agentes atuarão ou perceberão a ocorrência de eventos. Por fim, na programação orientada à organização cada agente assume um papel ao longo de uma organização social, onde são especificadas a interação com outros agentes, suas funções na organização e as normas que cada agente deve seguir durante a execução de suas tarefas.

2.2.2.1 Programação orientada a agentes

A programação orientada a agentes, primeiramente discutida por Shoham (1993), se tornou uma abordagem para o desenvolvimento de sistemas complexos e adaptativos. Nesta abordagem, o problema é pensado de modo a definir os papéis e as ações a cada um dos agentes que compõem o sistema.

Uma possível forma de implementação de agentes é a adoção da arquitetura BDI (*Belief, Desire, Intention*), que possibilita aos agentes mecanismos deliberativos para tomada de decisões (RAO; GEORGEFF, 1991). Segundo Bordini et al. (2007) esta abordagem oferece uma maneira para o desenvolvimento de sistemas de acordo com crenças, desejos e intenções.

As crenças representam o estado informacional do agente, suas crenças sobre o mundo incluindo a si próprio, os outros agentes, o seu conhecimento sobre o ambiente. Os desejos representam o estado motivacional do agente e constituem objetivos ou situações que o agente pode realizar ou produzir, os quais determinam o curso das ações do agente. As intenções representam o estado deliberativo do agente. As intenções são uma forma que os agentes têm para alcançar seus desejos. As intenções são planos em execução, onde um plano representa uma sequência de ações a serem executadas.

A execução de um plano e a modificação das crenças é realizada a partir da ocorrência de eventos. Um evento define acontecimentos que podem iniciar ou parar a execução de um plano, causar modificações (adição/remoção) sobre suas crenças ou ser resultados de percepções sobre o ambiente. O evento pode ser interno, quando o agente altera seu estado mental (modificando crenças) ou quando inicia ou finaliza a execução de um plano; ou externo, provindo do próprio ambiente ao qual o agente está situado.

O mecanismo de deliberação executado por um agente na arquitetura BDI, em particular baseado no *framework* Jason (BORDINI et al., 2007), ocorre da seguinte forma (Figura 2): na ocorrência de um evento, uma função de seleção de eventos (*SE*) seleciona um

único evento, dentre uma lista; a função de seleção de planos (*SO*) seleciona um plano que é aplicável a partir de um conjunto de planos disponíveis; e uma terceira função (*SI*) seleciona uma intenção em particular dentre todo o conjunto de intenções.

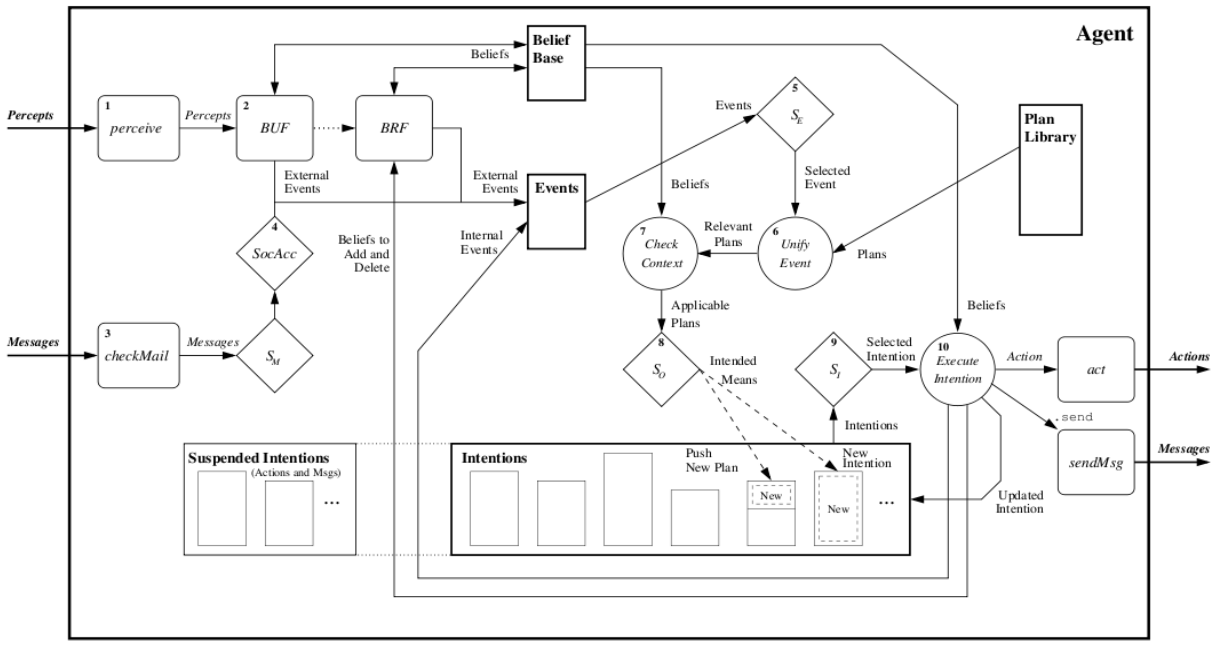

Figura 2: Mecanismo de deliberação do interpretador AgentSpeak.
Fonte: Bordini et al. (2007).

A cada ciclo de raciocínio de um agente, uma lista de eventos (*Events* na Figura 2) (internos ou externos) é atualizada. Depois da função *SE* ter selecionado um evento, o agente unifica o evento (*Unify Event*) que estão de acordo com o contexto do plano (*Unify Context)*, e por isso podem ser aplicados (*Applicable Plans*). Então a função *SO* seleciona um único plano aplicável do conjunto, que se torna uma intenção (*New Intentions, Push Subplan*). A função *SI* seleciona uma intenção (um plano entre os contidos na pilha de planos (*Intentions*) parcialmente instanciados). Esta intenção ao ser executada (*Action*) pode alterar o ambiente, alterar o estado mental, ou mesmo desencadear um novo evento.

Depois de executada a intenção, o ciclo de raciocínio, inicia-se novamente. Caso nenhum evento tenha ocorrido durante o ciclo, o agente fica temporariamente inativo. Ou seja, somente após a ocorrência de um evento o agente executará novas ações.

Nesse trabalho os componentes da arquitetura (os agentes) são especificados de acordo com o modelo de agentes BDI.

2.2.2.2 Programação orientada ao ambiente

Na programação orientada ao ambiente, o ambiente não é somente o elemento que sofre as ações e provê percepções para os agentes. O ambiente é considerado como parte interna ao sistema multiagente, de maneira diversa da abordagem clássica da Inteligência Artificial em que o ambiente é um componente externo (RICCI; PIUNTI; VIROLI, 2010).

Segundo Ricci et al. (2008), o ambiente deve apresentar as seguintes características:

(i) um nível básico que provê acesso aos recursos externos de *hardware/software* (sensores e atuadores, impressora, rede, banco de dados, serviços Web e outro recursos) com os quais interage o sistema multiagente;

(ii) também deve servir como uma ponte conceitual entre o nível de abstração dos agentes e os níveis mais básicos tornando detalhes de implementação transparentes aos agentes;

(iii) por fim, deve intermediar o acesso entre os agentes e os recursos.

2.2.2.3 Programação orientada à organização

A programação orientada à organização em sistemas multiagentes é uma abordagem inspirada em organizações baseadas em sociedades humanas, que adotam regras e papéis sociais aplicadas a soluções de problemas. Neste contexto, a organização é um conjunto de restrições que um grupo de agentes adota, de modo a controlar seu comportamento e facilitar a obtenção de objetivos globais. Nestas organizações o agente pode se comportar de maneira flexível, ou rígida, dependendo da política implantada na organização (HÜBNER et al., 2006).

As organizações seguem modelos comportamentais que podem ser divididos em duas classes: centrados em agentes ou centrados na organização. No primeiro caso, os agentes são os elementos que garantem a formação da organização, enquanto no segundo, a organização existe inicialmente com estrutura, regras e relações entre papéis. Nesta segunda abordagem, os agentes tendem a entrar e participar da organização.

Existem também modelos organizacionais que enfatizam planos globais da sociedade (PYNADATH; TAMBE, 2003). Neste caso, a preocupação é o funcionamento da organização, a partir da especificação dos planos globais, das políticas de atribuição de tarefas

aos agentes, da coordenação para executar um plano, e da qualidade (consumo de tempo, o uso de recursos, etc) de um plano.

A especificação organizacional de um sistema multiagentes (SMA) é útil para melhorar a eficiência do sistema uma vez que a organização limita os comportamentos de agentes para aqueles que são socialmente destinados. Isto significa que cada agente deve atuar de modo a permitir que a organização possa atingir um objetivo comum (GASSER, 2001). Dessa forma, os agentes buscam cooperar de modo que este objetivo possa ser alcançado. Nos SMA que não contam com um certo grau de organização, a autonomia dos agentes pode levar o sistema a um comportamento desordenado.

2.2.3 Comunicação em sistemas multiagentes

As arquiteturas baseadas em agentes necessitam definir modelos de comunicação entre os agentes, permitindo aos agentes que constituem o sistema trocarem conhecimentos ou solicitarem a execução de determinadas ações. Estes modelos incluem o uso de ontologias que definem um vocabulário de um domínio particular e permitem especificar a forma com que o conhecimento deve ser interpretado. Além disso, especificam um conjunto de protocolos de comunicação que são padronizados e independentes da linguagem de implementação dos agentes. Estes protocolos especificam a forma com que os agentes podem interagir para troca de recursos e a prestação de serviços, isto é, a sequência de mensagens que devem ser trocadas para uma determinada finalidade.

A KQML (*Knowledge Query and Manipulation Language*) (FINN et al., 1994) é uma linguagem de comunicação entre agentes resultante dos esforços da DARPA KSE (*Knowledge Sharing Effort*). A KQML tem por objetivo prover um protocolo para a troca de conhecimento entre os agentes. Além disso, KQML especifica uma sintaxe para um conjunto de diferentes mensagens e performativas.

As performativas definem qual a intenção do agente na mensagem enviada. Assim, um agente é capaz de entender o que outro está querendo com a mensagem enviada, por exemplo, se o mesmo está solicitando ou fornecendo uma informação.

Algumas performativas definidas da linguagem KQML são dadas a seguir:

- Performativas básicas:

  ask-if: o agente solicita a outro agente o valor verdade de uma requisição.

  ask-one: o agente solicita uma única resposta para uma dada requisição.

ask-all: o agente deseja saber todas as possíveis respostas de uma dada requisição.

- Performativas genéricas de informação

tell: o agente envia uma requisição a outro agente, que acredita ser verdadeira.

achieve: o agente envia uma requisição de um desejo ou objetivo que deve ser alcançado.

untell: o agente envia uma requisição a outro agente, que acredita ser falsa.

unachieve: o agente envia uma requisição de um desejo ou objetivo que deve ser abortado.

Existem várias outras performativas associadas à operações de rede, ações, requisições/ respostas entre outros. A seguir é discutido um exemplo de mensagem enviada por KQML (HÜBNER et al., 2006):

```
(ask-one
        :content  (PRICE IBM ?price)
        :receiver stock-server
        :language Lprolog
        :ontology NYSE-TICKS
)
```

A mensagem acima pode ser interpretada como uma requisição de um agente que solicita ao agente "stock-server" qual o preço de um produto IBM, usando a linguagem "Lprolog", que deve ser interpretada ou respondida segundo o vocabulário definido na ontologia "NYSE-TICKS". Uma possível resposta dada pelo agente "stock-server" seria:

```
(tell
        :content (PRICE IBM 2.000)
        :receiver client
        :language Lprolog
        :ontology NYSE-TICKS
)
```

A linguagem KQML é utilizada em vários *frameworks* que utilizam programação multiagentes, sendo Jason, um exemplo desses *frameworks*.

2.2.4 *Framework* para programação Multiagente - JaCaMo

O *framework* JaCaMo[1] é constituído pela união de três diferentes *frameworks* denominados Jason[2], para programação orientada a agentes cognitivos, CArtAgO[3], para a programação orientada ao ambiente, e Moise[4], para programação orientada à organização. Deste modo, JaCaMo apresenta-se como uma ferramenta que contempla os principais componentes da programação multiagentes.

Esse *framework* é utilizado como base para a especificação da arquitetura e também para a implementação da aplicação desenvolvida no estudo de caso.

2.2.4.1 Jason

Jason é um *framework* escrito em Java com as funcionalidades para a programação de agentes cognitivos. A linguagem interpretada na plataforma Jason é uma extensão do *AgentSpeak,* linguagem introduzida em Rao (1996), para programação de agentes baseados na arquitetura BDI. Além disso, uma característica interessante oferecida por esse *framework* é a possibilidade do sistema multiagente poder ser executado em um ambiente distribuído, usando como base a plataforma de agentes Jade[5].

Um agente implementado em Jason é constituído de uma base de crenças e um conjunto de planos. A base de crenças do agente é definida a partir de um conjunto de literais similares aos utilizados no paradigma lógico de programação. Isso significa que a base de crenças é constituída por um conjunto (vazio ou não) de informações simbólicas (predicados) relacionando propriedades e valores. Um exemplo de predicado é: "cor(vermelho)". Neste caso, o agente acredita que uma propriedade "cor" apresenta um valor "vermelho".

Assim como ocorre na linguagem do paradigma lógico, os agentes Jason podem, além de consultar e validar o valor verdade das crenças, também realizar o processo de unificação (BORDINI; HÜBNER; VIERA, 2005). Além disso, é possível criar regras mais complexas de inferências relacionando diversas crenças a partir de conectivos lógicos.

Os planos em Jason são definidos em duas categorias: planos de satisfação de objetivos e planos de testes. Os planos para satisfação de objetivos são definidos como planos onde os agentes realizam um conjunto de atividades ou novos subplanos. Enquanto que planos de

1 http://jacamo.sourceforge.net/

2 http://jason.sourceforge.net/wp/

3 http://cartago.sourceforge.net/

4 http://moise.sourceforge.net/

5 http://jade.tilab.com/

testes, quando executados tendem a unificar o valor de uma variável ou verificar seu valor verdade. Planos de testes são semelhantes à execução de consultas.

Em ambos os casos a definição estrutural de um plano é dada segundo três componentes: um evento, um contexto e o corpo. Um evento pode ser do tipo adição/deleção de um objetivo ou de uma crença. Contexto define possíveis condições sob as quais um plano será executado, ou servem para unificar valores de crenças que serão atualizadas ou removidas da base de crença do agente. Por fim, o corpo de um plano pode apresentar: adição/atualização/remoção de crenças, execução de novos objetivos, ou subobjetivos, ações de comunicação entre agentes e operações internas ao agente ou sobre artefatos.

As performativas utilizadas na comunicação entre os agentes na plataforma Jason são muito similares à linguagem de comunicação KQML e podem ser utilizadas junto à ação "send" definida previamente na API do próprio *framework*.

2.2.4.2 CArtAgO

CArtAgO (*Common ARtifact infrastructure for AGent Open environment*) é um *framework* desenvolvido na linguagem Java, que permite a criação de entidades computacionais denominadas artefatos. A noção de artefato no contexto da programação multiagente, consiste na forma que o desenvolvedor irá estruturar e organizar o ambiente. Neste sentido, o ambiente é concebido como um conjunto dinâmico de artefatos representando recursos e ferramentas que os agentes podem compartilhar, explorar e trabalhar no mesmo ambiente. Além disso, o conjunto global de artefatos pode ser organizado em uma ou múltiplas áreas de trabalho, possivelmente distribuídas em diferentes nós de uma infraestrutura de rede (RICCI et al., 2008).

As funcionalidades dos artefatos não estão restritas ao uso a partir dos agentes do sistema, podendo ser compartilhada com agentes de outros sistemas. Já do ponto de vista do agente, os artefatos são as entidades funcionais que compõem o ambiente em que estão situados.

Um artefato, conforme ilustrado na Figura 3, é constituído por um conjunto de operações e um conjunto de propriedades observáveis.

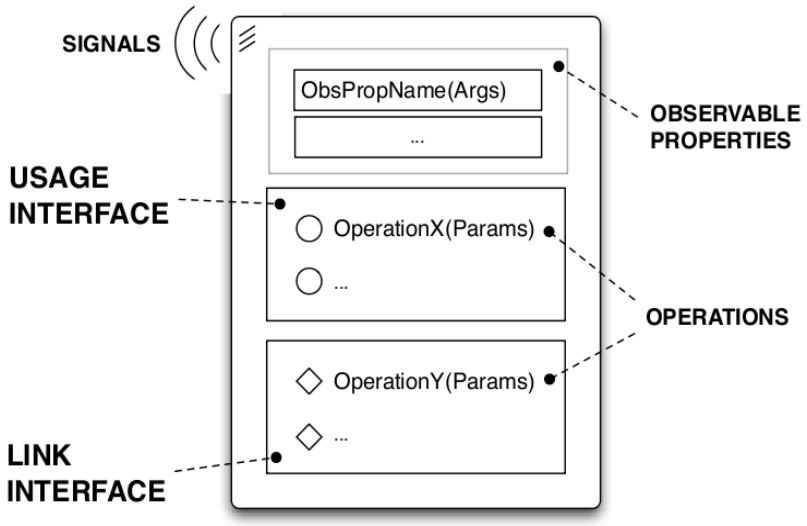

Figura 3: Representação de um artefato em programação orientada ao ambiente.
Fonte: Ricci et al. (2010).

As operações (*Operations*) representam ações executadas pelos artefatos, similar ao conceito de métodos. Existem dois tipos de operações disponíveis como operações internas ao artefato (*Usage Interface*), que alteram o estado das variáveis do próprio artefato ou externas a ele, relacionando dois ou mais artefatos diferentes (*Link Interface*). As propriedades observáveis (*Observable Properties*) representam variáveis do artefato e são acessíveis aos agentes. Além disso, a execução de uma operação pode também gerar sinais (*Signals*) que são percebidos pelos agentes.

A programação das operações é definida a partir de anotações Java e métodos herdados. As propriedades observáveis são definidas por meio da execução do método *defineObsProp* que relaciona uma variável interna da classe à um nome, que será utilizado como o nome da propriedade observável pelos agentes. Os sinais podem ser definidos pelo método "signal" que define um nome que será utilizado pelos agentes para percepção de sinais emitidos pelo artefato.

A integração dos *frameworks* Jason e CArtAgO é denominada JaCa e apresenta uma série de extensões que visam integrar diferentes tecnologias, e plataformas no desenvolvimento de ambiente para agentes.

2.2.4.3 Moise

O Moise (*Model of Organisation for multI-agent SystEms*) é um *framework* escrito em Java que permite a programação orientada à organização de sistemas multiagentes. Os três conceitos principais usados para construir os diferentes níveis de uma organização são os seguintes: papéis, relações de função e grupos .

O Moise está estruturado em três níveis: no nível estrutural são definidas a estrutura e relação entre os papéis; no nível funcional é definido a função dos papéis na execução de planos globais do sistema; e, no nível normativo é definido as regras e as restrições de cada papel em relação à organização. A especificação estrutural da organização pode ser descrita a partir de uma linguagem baseada em diagramas denominada OML (*Organisation Modelling Language*) (HÜBNER; SICHMAN; BOISSIER, 2004).

O nível estrutural é definido segundo os conceitos denominados papel, grupo e suas relações. Um grupo é formado por um conjunto de papéis organizados segundo ações e interesses comuns. Cada papel e grupo é estruturado segundo as seguintes relações que podem ser inter ou intra-grupos:

- Informação - *acquaintace (acq)*: define uma relação em que um papel deve informar sua representação do mundo para outro papel.
- Comunicação – *communication (com)*: define uma relação em que existe permissão de um papel se comunicar com outro.
- Autoridade – *authority (aut)*: define uma relação que permite que um papel possa controlar as ações de outro papel. Este tipo de relação implica na relação de informação ou comunicação em sentido inverso.
- Compatibilidade – *compatibility (compat)*: define uma relação em que dois papéis são compatíveis segundo sua definição estrutural e funcional.

Além das relações já discutidas, o nível estrutural ainda define papéis abstratos, ou seja, papéis que não podem ser interpretados por nenhum agente, e somente têm função de facilitar a estruturação da organização.

O nível funcional é definido segundo os conceitos de missão (um conjunto de metas) e planos globais. Estes dois conceitos são a base para a definição em um esquema Social *(Social Scheme*), que é essencialmente uma árvore de decomposição de metas onde a raiz é o objetivo global da organização e onde as responsabilidades de cada agente para os subobjetivos são distribuídos em missões (HÜBNER; SICHMAN; BOISSIER, 2004). Cada meta pode ser decomposta em subobjetivos através de planos que podem utilizar três operadores, que são apresentados no exemplo na Figura 6:

- Sequência (,): o plano de "g2 = g6, g9" significa que o objetivo g2 será alcançado se o objetivo g6 é alcançado e depois também o g9 for alcançado.
- Escolha (|): o plano "g9 = g7 | g8" significa que o objetivo g9 objetivo será alcançado se um dos objetivos, g7 ou g8 for alcançado;
- Paralelismo (||): o plano "g14 = g18||g19" significa que o objetivo g14 será alcançado se ambos g18 e g19 são alcançados, mas que podem ser executados em paralelo.

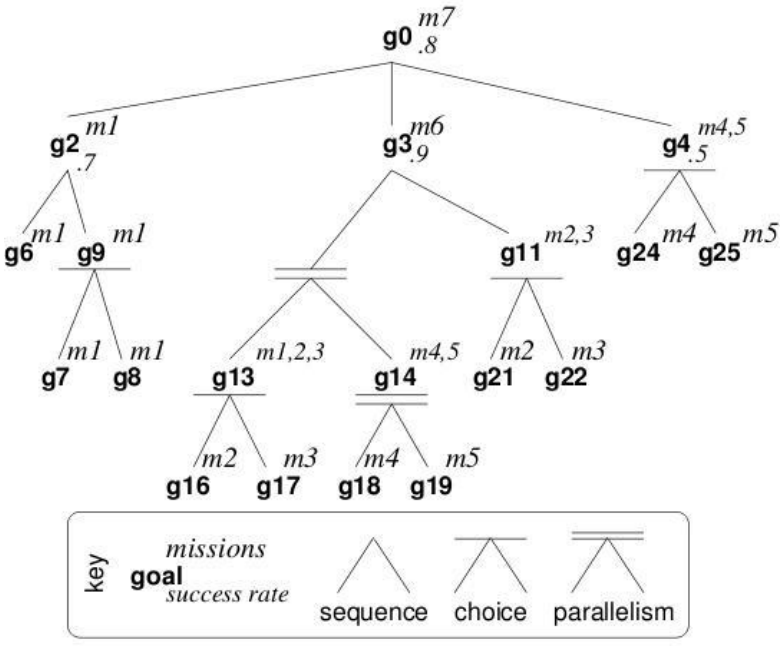

Figura 4: Especificação funcional segundo a linguagem OML.
Fonte: Hübner, Sichman e Boissier, (2002).

O nível normativo é especificado por dois tipos de relações, denominadas obrigação e permissão. Estas duas relações determinam o comportamento de um papel frente a execução de uma missão. A relação de obrigação determina que um papel é obrigado a executar uma determinada missão. Já na relação de permissão, o agente que desempenha o papel pode estabelecer o melhor momento para executar dependendo do interesse da própria organização.

No *framework* JaCaMo os três níveis da organização são implementados a partir de artefatos organizacionais. O conjunto destes artefatos cria uma infraestrutura denominada ORA4MAS (HÜBNER et al., 2010).

2.3 Arquitetura Orientada a Serviços

Arquitetura Orientada a Serviços (SOA – *Service-Oriented Architecture*) consiste em uma abordagem que especifica infraestruturas para o desenvolvimento de aplicações, onde os diversos recursos de um sistema são encapsulados e disponibilizados através de interfaces de comunicação padronizadas, sendo baseadas principalmente na linguagem XML. Esta abordagem garante a independência de plataforma e da linguagem de programação em que os sistemas foram desenvolvidos. Dessa forma assegura a interoperabilidade entre diferentes sistemas.

Segundo Cândido et al. (2010), SOA estabelece um modelo arquitetural que tem como foco garantir eficiência, agilidade e produtividade ao colocar serviços como meios primários a partir do qual uma solução é representada. Neste contexto, existem duas principais especificações para a implementação de sistemas baseados em SOA (PAUTASSO; ZIMMERMANN; LEYMANN, 2008): Serviços Web definido pelo W3C (MCCABE et al, 2004.) e Serviços REST (*Representational State Transfer*).

Apesar de serem robustas e testadas largamente, estas especificações não se encaixam em todos os tipos de aplicações e exibem algumas restrições relacionadas à complexidade arquitetural de seus modelos. Além disso, nem sempre é possível a implementação dessas especificações em sistemas embarcados e no contexto da IoT. Segundo Shen et al. (2010), o uso de tecnologias baseadas em serviços Web em dispositivos com restrições de recursos pode se tornar possível a partir da adoção de especificações enxutas. As duas principais especificações de serviços no contexto da IoT são o *Devices Profile for Web Services* (DPWS) e o *OPC Unified Architecture* (OPC-UA) (CANDIDO ET AL., 2010).

O DPWS é uma especificação aberta de SOA, definida de acordo com a especificação de serviços Web, que vem sendo usada para a interoperabilidade de dispositivos/sistemas embarcados (DOHNDORF, 2010). A OPC-UA é uma especificação fechada de SOA, definida de acordo com os serviços REST. Neste trabalho foi adotado o DPWS por tratar-se de uma especificação aberta.

2.3.1 Serviços Web

De acordo com Papazoglou (2003), serviços Web são caraterizados por sistemas onde sua funcionalidade é exposta através de uma interface que permite descrição, localização e chamada a partir de um formato padronizado. Serviços Web são definidos de modo a

transformar a Web orientada em documentos em um sistema orientado a serviços onde as máquinas são capazes de acessar fontes de dados.

Dentro da visão de SOA, as aplicações implementadas com base nas especificações de serviços Web estão organizadas segundo três componentes principais (Figura 5): Provedor - representa os serviços, juntamente com as entidades responsáveis por criar, publicar e disponibilizá-los; Cliente - representa quem consome os serviços (humanos ou maquinas); e, Registro de Serviços - representa um repositório que centraliza a descrição dos serviços disponíveis. O repositório do Registro de Serviços é utilizado pelos Provedores para publicar (Publicação) as descrições dos serviços, enquanto os Clientes o utilizam para buscar (Busca) por serviços. Após encontrar um serviço, o Cliente através da descrição do mesmo, é capaz de estabelecer uma conexão (Conexão) com o Provedor de serviço e consumir os serviços oferecidos.

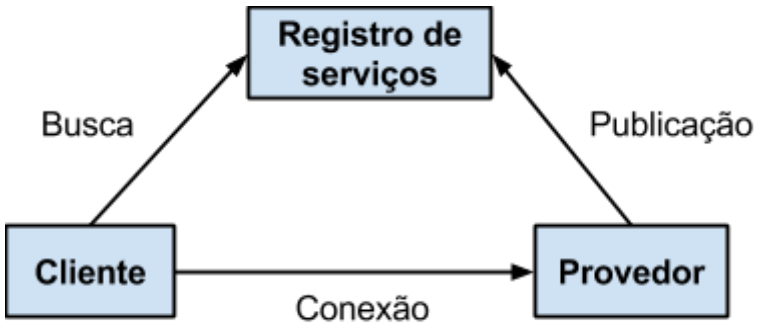

Figura 5: Principais componentes de arquiteturas orientadas a serviço.

O padrão de serviços Web definido pelo W3C é baseado na linguagem XML, o que permite a integração e troca de mensagens entre os componentes (DIAS; REGIS, 2012). Neste padrão, a comunicação (conexão da Figura 8) é baseada no protocolo SOAP (*Simple Object Access Protocol*), um padrão aberto e suportado por várias empresas de TI (Sun, Microsoft, HP, IBM, Oracle, BEA). O SOAP adiciona uma camada sobre o nível de aplicação na pilha TCP/IP, de tal forma que as mensagens podem ser transportadas por qualquer protocolo de aplicação de Internet (HTTP, SMTP, FTP, etc.) (BAGNASCO et al., 2006).

A descrição dos serviços (operações, tipos de dados e como acessar o serviço) é feita utilizando a linguagem de descrição de serviços WSDL (*Web Service Description Language*). A publicação, pesquisa e descoberta do serviço é realizada por meio de serviços de registro que utilizam a infraestrutura UDDI (*Universal Description, Discovery and Integration*).

A tecnologia de serviços Web é atualmente a solução de software mais utilizada para a implementação de SOA ao longo das diferentes camadas do uso de sistemas em ambientes de IoT. Segundo Aziz et al. (2004), estas abordagens baseadas em serviços adotam uma arquitetura em duas camadas, onde a camada de mais baixo nível (nível dos dispositivos) provê serviços básicos para acessar recursos de hardware heterogêneos, e uma camada de mais alto nível faz o uso dos dados disponibilizados pela camada mais baixa.

2.3.2 Padrão de Serviços para dispositivos - DPWS

A adoção de serviços baseados em SOAP em dispositivos embarcados, muito utilizada como recurso para sistemas de automação, é limitada pela redundância da comunicação e o excesso de processamento necessário para análise das mensagens (CÂNDIDO et al., 2010). Entretanto, algumas tentativas de portar SOAP em dispositivos embarcados existem na literatura. Entre tais abordagens destacam-se o g-SOAP (VAN ENGELEN; GALLIVAN, 2002), eSOAP (KAKANAKOV; SPASOV, 2005) e o kSOAP[6] (KOBJECTS.ORG, 2006). Além dessas, duas principais especificações surgiram: o *Devices Profile for Web Services* (DPWS), e o *OPC Unified Architecture* (OPC-UA) (CANDIDO ET AL., 2010), que compreende um conjunto de padrões que permite dispositivos embarcados executar os serviços Web de forma nativa (SUCIC; BONY; GUISE, 2012).

O DPWS é a especificação de *middleware* que define dois elementos fundamentais: o dispositivo e seus serviços internos (IZAGUIRRE; LOBOV; LASTRA, 2011). O DPWS também especifica uma infraestrutura de serviços previamente construídos:

- Descoberta de serviços (*WS-Discovery*): usado por um dispositivo conectado a rede para informar aos demais sobre a sua entrada no ambiente e descobrir outros serviços e dispositivos.
- Serviços para troca de metadados (*WS-MetadataExchange*): provê acesso a serviços dos dispositivos e aos seus metadados (WSDL e XML Schema).
- Serviços de publicação e assinatura (*publish/subscribe)* para eventos (*WS-Eventing*): permitem a assinatura, por parte do clientes, ou publicação, por parte dos dispositivos de serviços executados a partir eventos.

[6] http://kobjects.org/

O DPWS é construído sobre o padrão SOAP 1.2 e reside em especificações adicionais de serviços, tais como *WS-Addressing* e *WS-Policy*.

As mensagens são enviadas usando os protocolos SOAP 1.2.

O modelo de operação mais comum do DPWS consiste na descoberta de dispositivos relevantes na rede, obtenção da descrição dos dispositivos e os serviços que os mesmos proveem. Portanto o DPWS apresenta uma pequena e eficiente infraestrutura para conexões entre dispositivos, totalmente compatível com as especificações de serviços Web.

Existem diversos *frameworks* para dar suporte à implementação de infraestruturas de serviços DPWS. Um exemplo é o *framework* chamado JMEDS[7] uma API Java que implementa a especificação do DPWS.

2.4 Web Semântica

A Web Semântica é uma iniciativa liderada pelo W3C (*World Wide Web Consortium*) como uma extensão da Web atual com o objetivo de criar um meio universal para a troca de informação, atribuindo significado (semântica) ao conteúdo dos documentos da Web, de modo que o significado do conteúdo seja compreendido não só por humanos, mas também por máquinas (BERNERS-LEE; HENDLER; LASSILA, 2001). Para alcançar esse objetivo o W3C propõe o uso de um conjunto de tecnologias, entre as quais se encontra as ontologias.

Nesse trabalho, é proposto o uso de tecnologias da Web Semântica, em especial de ontologias, para a descrição semântica dos dispositivos e seus recursos, bem como dos processos para o gerenciamento destes dispositivos. Além disso, as ontologias também definem o vocabulário trocado pelos agentes do sistema. Com isso pretende-se obter uma melhor integração e processamento dos dados, bem como a automação da execução de tarefas do sistema, definindo os processos e recursos relacionados (ABOWD et al., 1999).

2.4.1 Ontologias

Uma ontologia é utilizada para representar o conhecimento de uma área de interesse através da definição dos conceitos e das relações entre os mesmos (SONG; CÁRDENAS;

[7] http://ws4d.e-technik.uni-rostock.de/jmeds/

MASUOKA, 2010). De acordo com o contexto e generalidade, é possível classificar a ontologia em tipos:

- As ontologias gerais, que descrevem conceitos mais genéricos e independentes de um domínio;
- As ontologias de domínio, que descrevem o vocabulário relativo a um domínio específico por meio da especialização dos termos das ontologias gerais;
- As ontologias de processos, que descrevem o vocabulário relativo a uma tarefa ou atividade genérica, utilizando para isso conceitos oriundos das ontologias de alto nível e de domínio.
- As ontologias de aplicação, que são as mais específicas e são construídas sobre os conceitos das demais ontologias e descrevem conceitos utilizados em uma aplicação.

A construção de uma ontologia ocorre a partir da definição de conceitos, sua taxonomia, possíveis relações e axiomas. Um conceito representa uma categoria de coisas similares que compartilham um conjunto de propriedades comuns. Uma relação expressa a associação entre conceitos ou entre um conceito e um determinado valor. A taxonomia entre classes é definida pelo uso de relações de subclasse. Por fim, axiomas são afirmações lógicas construídas com as demais entidades que formam as ontologias. (BATRES et al., 2005).

Segundo Jasper e Uschold (1999), as três aplicações mais comuns de ontologias ocorrem na comunicação entre humanos e máquinas; na capacidade de prover interoperabilidade entre sistemas, permitindo a definição e o uso de conceitos comuns; e, no desenvolvimento de sistemas de maior qualidade, garantindo uma melhor expressividade semântica para os conceitos manipulados.

Muitos dos problemas relacionados a IoT citados anteriormente, decorrentes da falta de padronização entre esquemas de dados e heterogeneidade de dispositivos são causados pela falta de padronização de conceitos e terminologias associadas à aplicações. Portanto, uma forma de prover interoperabilidade é a adoção de ontologias para definições de conceitos comuns.

As ontologias podem prover uma forma para representação de informações relevantes acerca dos conceitos relacionados ao domínio de aplicações suportando ao compartilhamento e representação formal do conhecimento utilizado nessas aplicações. Tais aplicações de ontologias em ambientes da Internet das Coisas têm contribuído para o paradigma de desenvolvimento de aplicações denominado Web Semântica das Coisas.

2.4.2 Web Semântica das Coisas

A Web Semântica das Coisas (*Semantic Web of Things* - SWOT) é uma visão que une tecnologias da Web Semântica e a Internet das Coisas. O objetivo da SWOT é integrar informações semânticas aos dispositivos do mundo físico, de modo a conectar dispositivos, usuários e diferentes entidades digitais. A SWOT tem impacto significativo nos modelos de interação humano-computador (mais geralmente, homem-dispositivo), pois reduz o esforço dos usuários na extração de informações, a partir de informações contextuais extraídas a partir dos dispositivos (RUTA et al., 2013).

Segundo Abowd et al. (1999), contextos podem ser definidos como qualquer informação usada para caracterizar a situação de uma entidade. Uma entidade pode ser uma pessoa, lugar, tempo ou objeto, ou seja, tudo que é considerado relevante para execução de ações ou interações entre usuários e sistemas. Isso significa que um sistema com tais características deve ser capaz de determinar características de entidades (quem, o quê, onde, quando, por que) com o mínimo de informação necessária.

As tecnologias da Web Semântica, como ontologias, têm sido usadas para representação e compartilhamento de conhecimento, melhorando a integração de dados e a interoperabilidade de sistemas. Ontologias são usadas para anotar semanticamente os serviços fornecidos por dispositivos, a fim de melhorar e automatizar a sua descoberta, acesso e composição. Neste sentido, ontologias de domínio e aplicação devem ser especificadas para definir um vocabulário comum que descreve os conceitos de domínio, tais como dispositivos, esquemas de dados, eventos, monitoramento e controle de tarefas (SONG; CARDENAS; MASUOKA, 2010). Há várias ontologias desenvolvidas para diferentes domínios que podem ser utilizadas e adaptadas. A anotação semântica dos dispositivos e seus recursos permite que agentes de software possam encontrá-los e usá-los para automatizar suas tarefas.

A fim de estabelecer um padrão de interoperabilidade semântica entre diferentes dispositivos em redes de sensores, o grupo W3C Semantic Sensor Network Incubator (SSN-XG), desenvolveu uma ontologia denominada Semantic Sensor Network Ontology (SSN). Essa ontologia define um vocabulário para descrever sensores em termos dos métodos de medição, observações e suas relações (COMPTON et al., 2012).

3 TRABALHOS RELACIONADOS

A partir dos trabalhos relacionados selecionados na pesquisa bibliográfica foram analisados e identificados os principais requisitos e as tecnologias a serem utilizadas para especificar uma arquitetura para o gerenciamento de dispositivos em ambientes baseados em IoT. Nesse estudo buscou-se também identificar como tem sido tratada as formas utilizadas para a interoperabilidade de dispositivos heterogêneos bem como a integração e análise de dados. Outras questões investigadas são referentes às abordagens utilizadas para o desenvolvimento de aplicações para o gerenciamento de ambientes baseados em IoT.

Nesse estudo é dado atenção especial para trabalhos que utilizam as abordagens de SOA, Sistemas Multiagentes e Web Semântica no contexto da IoT.

3.1 Interoperabilidade de dispositivos heterogêneos em IoT

No contexto de hardware e recursos de computação, as "coisas" (dispositivos) necessitam se conectar à Internet e serem capazes de enviar e receber informações de eventos relacionados ao ambiente em que estão inseridos. Além disso, estes dispositivos devem ser capazes de se conectar automaticamente, sem a necessidade de intervenção humana. Dessa forma, torna-se necessário o uso de tecnologias de interoperabilidade, visando a criação de modelos de dados comuns a fim de proporcionar a descoberta, o acesso, e a publicação de serviços.

Neste sentido, Kosovac (2007) apresenta uma infraestrutura baseada em serviços Web para gerenciar informações em ambientes heterogêneos, distribuídos e autônomos, onde a execução automática de tarefas pode ser realizada segundo as necessidades do usuário. A adoção desses serviços como uma infraestrutura de comunicação assegura ao sistema o uso de um padrão aberto e independente de linguagens.

Solomon e Ionescu (2007) também apresentam o uso de serviços em uma arquitetura para sistemas em tempo real. Nesta arquitetura, o sistema é decomposto em elementos básicos como sensores, atuadores, controladores entre outros. Cada elemento provê seus dados e ações na forma de serviços. Além disso, os Serviços Web especificados fazem o uso de um registro de serviços (UDDI), que permite a descoberta e a conexão a novos componentes.

Uma alternativa para o uso de registros de serviços é apresentada no trabalho de Stirbu (2008), onde o uso de serviços integra funcionalidades *Plug&Play* para redes de sensores e

atuadores heterogêneos. Essas funcionalidades fazem com que os dispositivos ao se conectarem na rede anunciem seus recursos e com isso os outros componentes da rede são capazes de identificá-los e utilizá-los de forma automática. Neste caso, os serviços são baseados em princípios RESTful que junto aos elementos da Web Semântica oferecem um mecanismo de descoberta de novos dispositivos.

Apesar de Serviços Web permitirem a interoperabilidade de comunicação entre sistemas e componentes heterogêneos, as complexidades inerentes da tecnologia de serviços Web, como SOAP, XML e HTTP é demasiada elevada para o uso em ambientes frequentemente encontrados em aplicações máquina-a-máquina. Isso significa que soluções mais leves são necessárias para determinadas aplicações, uma solução possível é o uso da especificação DPWS (IZAGUIRRE; LOBOV; LASTRA, 2011).

Em Moeller e Sleman (2008) é apresentado o uso do DPWS para integrar Serviços Web, permitindo que redes de sensores sem fio possam disponibilizar dados para outras redes baseadas em TCP/IP. Neste caso, a tecnologia DPWS serve como um *gateway,* expondo os dados coletados a partir dos sensores da rede sem fio, na forma de serviços.

O trabalho de Cucinotta et al. (2009), adota ambas especificações (DPWS e Serviços Web), em uma arquitetura SOA para automação industrial que visa tratar dispositivos e subsistemas em diferentes escalas e tempos de atuação. A arquitetura é desenvolvida em duas camadas, a camada com os Serviços Web específicos para expor para a rede as variáveis utilizadas em processos de monitoramento e na outra camada as funções para a operação dos dispositivos. O padrão DPWS é utilizado para permitir funcionalidade do tipo *Plug&Play* para novos dispositivos adicionados à arquitetura.

Dias e Regis (2012) apresentam o desenvolvimento de uma plataforma usando arquitetura (SOA) implementada a partir do padrão DPWS para a integração entre dispositivos legados e os sistemas de monitoramento e supervisão para processos industriais. Estes serviços foram modelados de modo a prover um *driver* genérico para dispositivos controladores legados.

A otimização do protocolo DPWS é trabalhada em Samaras et al. (2013) a partir de uma modificação da pilha de protocolos aplicado em redes de sensores sem fio (RSSF) e com baixo consumo de energia (6LoWPAN). A modificação é baseada em um novo formato para as trocas de mensagens DPWS. Este protocolo modificado apresentou menores tempos de processamento e utilizou menos recursos computacionais em comparação com as implementações padrão.

Por fim, Han et al. (2014 a) apresentam um kit de ferramentas de simulação, denominado DPWSim, para apoiar o desenvolvimento de aplicações para IoT. DPWSim permite aos desenvolvedores a criação de protótipos, desenvolvendo e testando os aplicativos da Internet das Coisas usando a tecnologia DPWS sem a presença de dispositivos físicos.

Esses trabalhos mostram a importância do papel de tecnologias baseadas em SOA para permitir não apenas a interoperabilidade entre componentes heterogêneos e distribuídos, mas também prover infraestruturas para a publicação, descoberta e acesso a esses componentes e seus recursos.

3.2 Inteligência Ambiental

Os dispositivos heterogêneos, presentes em ambientes da IoT, além de disponibilizar dados e realizar ações, também devem realizar tarefas para os usuários de maneira autônoma de acordo com sua necessidade (STAVROPOULOS et al., 2013). Neste caso, interfaces específicas são necessárias para facilitar o controle tanto dos dispositivos quanto das características do ambiente.

Neste sentido, Rohokale et al. (2011) apresentam uma abordagem cooperativa para monitorar e controlar a saúde de seres humanos. Na abordagem cada ser humano usa um dispositivo RFID, conectado à uma rede de sensores sem-fio. Quando um dos parâmetros monitorados (como pressão sanguínea, nível de hemoglobina, etc) apresentar anormalidades, o centro de tratamento de saúde será automaticamente comunicado. Além disso, a abordagem visa otimizar o uso dos recursos dos dispositivos.

No trabalho de Cho, Kyung e Baek (2013) é apresentado um sistema de monitoramento de dispositivos composto de um aplicativo de *smartphone* e dispositivos de coleta de dados. Basicamente, o sistema de monitoramento busca obter dados a partir de um algoritmo rápido de coleta de dados dos sensores dos dispositivos. Os dispositivos, buscam aprender a partir de heurísticas, formas de ajustar seu planejamento e descobrir o tempo adequado para enviar os seus dados de maneira autônoma.

Stavropoulos et al. (2013) apresentam uma infraestrutura de serviços Web para estes ambientes, denominado aWESoME, visando garantir um acesso rápido, descoberta, invocação e a execução de serviços. A solução proposta utiliza padrões abertos da Web amplamente utilizados. A nível de hardware a aWESoME utiliza rede de sensores e atuadores sem fio. A nível de aplicação, diversos clientes exploram a infraestrutura em diferentes plataformas, com o objetivo de mostrar a sua utilidade e eficácia para o monitoramento e economia de energia.

Boyle, Yates e Yeatman (2013) apresentam um estudo de caso sobre as aplicações de redes de sensores e dispositivos inteligentes em diversas áreas da cidade de Londres realizado no ano de 2013. O estudo apresenta uma série de sistemas de monitoramento nas áreas de transporte, ambiente e pessoas, destacando seu uso, tecnologia e tempo de respostas.

No trabalho de Aloulou et al. (2013) é apresentada uma abordagem para desenvolver um sistema para monitoramento de ambientes, em uma casa de repouso, a partir do estado de saúde de pacientes apresentando demência. A solução é constituída por um conjunto de sensores e dispositivos controlados por uma plataforma de software. Os resultados da avaliação do desempenho do sistema mostram a evolução da precisão na detecção precoce da degradação das condições dos pacientes.

Murar e Brad (2014) apresentam uma arquitetura para monitoramento e controle de dispositivos inteligentes aplicada em ambientes industriais. A arquitetura integra vários dispositivos numa rede sem fio, disponibilizando seus dados e ações, a partir do protocolo TCP/IP para dispositivos móveis.

Com base nos trabalhos analisados é possível destacar a importância da necessidade de infraestruturas e aplicações para o monitoramento e também o controle de ambientes em diversos domínios. Além disso, também é possível notar a grande presença dos dispositivos inteligentes, sejam eles incorporando elementos sensores ou atuadores. Em muitos casos esses dispositivos precisam trabalhar em conjunto e de forma eficiente para otimizar a utilização dos recursos disponíveis. Nesse sentido, os sistemas de monitoramento e controle tem um papel importante na integração e processamento dos dados bem como no gerenciamento dos dispositivos. Para o desenvolvimento desses sistemas, abordagens que utilizam técnicas de Inteligência Artificial são amplamente empregadas.

3.3 Web Semântica das coisas

Outro aspecto importante apresentado por aplicações desenvolvidas em ambientes da IoT, é a identificação de contexto e anotação semântica dos recursos presentes no ambiente. Neste caso, uma possível abordagem se dá a partir do uso de ontologias.

Em Aloulou et al. (2013), ontologias são usadas para representar o conhecimento sobre as entidades em um ambiente de casa de repouso. Esse ambiente de colaboração envolve pacientes, cuidadores, serviços auxiliares, sensores e dispositivos de interface. O

comportamento do sistema é determinado pelo conhecimento definido pelas ontologias, onde são representados o conhecimento sobre as condições e necessidades do paciente.

O trabalho de Ruta et al. (2013) apresenta o uso de conteúdo semântico em ambientes cooperativos, onde os recursos podem ser descobertos, consultados e anotados por objetos autônomos em uma rede *peer-to-peer*. Isso permite que os recursos atuem de forma colaborativa, sem a necessidade de um controle central e de coordenação.

As ontologias também são usadas para descrever o contexto da aplicação, a fim de realizar a execução de serviços mais adequados em um determinado cenário. A sensibilidade ao contexto permite a modificação automática do comportamento do sistema de acordo com a situação atual com a mínima intervenção humana (KUKA; NICKLAS, 2014).

Em Han, Lee e Crespi (2014b) é apresentada uma arquitetura para o tratamento de dados adquiridos por meio de dispositivos. Estes dados são coordenados de acordo com o contexto do ambiente do usuário e as regras determinadas pelos diferentes contextos.

A partir destes trabalhos e também outros que podem ser encontrados na literatura, pode-se observar a adoção das tecnologias da Web Semântica para lidar com diversos aspectos em ambientes de IoT. Nesse sentido, o desenvolvimento e uso de ontologias têm um papel fundamental para a integração e o compartilhamento dos recursos e do conhecimento utilizado na aplicação. Essas tecnologias tornam possível a automação da execução de diversas tarefas, e também contribuem para a escalabilidade e adaptação do sistema.

3.4 Abordagens multiagentes para Ambientes de IoT

As características de sistemas complexos e distribuídos em um ambiente de IoT, torna mais difícil a gestão dos dispositivos. Essa complexidade surge a partir dos comportamentos emergentes que aparecem a partir das interações entre as partes constituintes. A abordagem de sistemas multiagentes é um paradigma que fornece um conjunto de abstrações adequadas para a concepção e o desenvolvimento de sistemas distribuídos e heterogêneos que envolvem interações complexas entre entidades. Neste caso, os SMA têm como objetivo garantir a organização e o gerenciamento dos dispositivos em aplicações de IoT (ANGULO-LOPEZ; JIMENEZ-PEREZ, 2012).

O trabalho de Ângulo-Lopez e Jimenez-Perez (2012) propõe um framework baseado em agentes colaborativos para permitir o desenvolvimento de aplicações em ambientes da IoT, que objetivam a execução de tarefas mais complexas utilizando recursos distribuídos. Os

componentes apresentados no trabalho para o desenvolvimento do trabalho apresentam as seguintes características: conhecimento, descoberta, tomada de decisão, processamento de dados, ações e aspectos de segurança.

Entretanto, no trabalho não é citado como ocorrerá a interação com outros agentes nem se há uma estrutura de organização social do SMA. Além disso, não é comentado nenhuma organização de papéis nem quais serão os cenários de execução da arquitetura proposta. O trabalho também é focado na resolução de questões e integração de diferentes domínios.

O trabalho de Pujolle (2006) apresenta uma arquitetura multiagentes autônoma para controlar as comunicações entre os dispositivos distribuídos através de equipamentos de rede. Pujolle propõe em sua arquitetura quatro camadas: dados, controle, conhecimento e gerenciamento. A camada de dados é responsável por receber e direcionar os pacotes de dados da rede. Esta camada recebe mensagens de configuração enviadas pela camada de controle, visando otimizar o fluxo de dados pela rede. A camada de conhecimento é responsável por buscar os algoritmos de otimização e informá-los às demais camadas. Por fim, a camada de gerenciamento é responsável pelas tomadas de decisão de todo o sistema.

Estas camadas são implementadas a partir de agentes cujas funcionalidades estão distribuídas em vários papéis organizados em dois grupos: os agentes cognitivos e os agentes reativos. Os agentes cognitivos, compõem as camadas de controle, conhecimento e gerenciamento, e têm a função de supervisão da execução das tarefas dos demais agentes. Os agentes reativos compõem a camada de dados sendo responsáveis pelas tarefas operacionais da arquitetura. Contudo, a arquitetura visa somente otimizar o número de pacotes que trafegam pela rede. Dessa forma, nenhuma ação é executa no ambiente físico e nenhum mecanismo de descoberta é apresentado. Também não há nenhuma identificação de contextos, isso significa que no trabalho o objetivo da camada cognitiva é selecionar algoritmos de otimização segundo o fluxo de dados na rede.

Em Fuentes-Fernández et al. (2009) é proposta uma arquitetura baseada no paradigma de sistemas multiagente, definindo a separação entre a camadas dos agentes do sistema e a camada composta por dispositivos. Estes agentes são capazes de comunicar e negociar serviços para alcançar os objetivos do sistema. As atividades são organizadas de acordo com os papéis de cada agente, permitindo ao sistema ações de integração de dados, gerenciamento de sensores, processamento de dados e adaptação às mudanças. A organização entre camadas e a separação de atividades em papéis, permitem o desacoplamento entre o gerenciamento dos dados e as atividades ao nível de rede.

Karnouskos e Tariq (2009) propõem uma abordagem de SMA para simular uma infraestrutura dinâmica e emergente de um grande número de dispositivos distribuídos, habilitados a prover recursos na forma de Serviços Web. Neste SMA um conjunto de agentes com funções diferentes são responsáveis por lidar com várias tarefas ao longo do sistema.

No trabalho é implementado um ambiente simulado de dispositivos utilizando como tecnologia de interoperabilidade a especificação de serviços DPWS. Na camada de sistema multiagente, cada um dos agentes é responsável por gerenciar unicamente um dispositivo conectando-se aos dispositivos via DPWS. Dessa forma, além de garantir a comunicação entre os agentes, o DPWS fornece as funcionalidades de detecção de novos dispositivos na rede local. Os agentes se comunicando com os dispositivos utilizam o DPWS para comunicação externa ao sistema e ACL para comunicação com os agentes do SMA. Os papéis dos agentes do SMA são:

- Agente de gerenciamento (*Management Agent*): associado às funções de gerenciamento e análise das requisições do usuário.
- Agente Explorador de dispositivos (*Device Explorer Agent*): agente responsável pela detecção dos dispositivos DPWS.
- Agente Criador de dispositivos (*Device Generator Agent*): agente criado pelo Agente Explorador de dispositivos para gerenciar a execução das operações dos dispositivos.
- Agente de cenário (*Scenario Agent*): criados especificamente para um dado cenário de acordo com as necessidades do mesmo.
- Agente de Serviço (*Service Agent*): agente que simula um dispostivo DPWS.

Contudo no trabalho não são tratadas as questões de cooperação entre os diversos agentes e dispositivos. Além disso os agentes de cenários são entidades especificadas para um dado contexto.

Leong et al. (2014) propõem uma abordagem Web multiagente para IoT. A arquitetura multiagente fornece características autônomas para IoT tornando seus componentes gerenciáveis. Além disso, a abordagem Web multiagente proposta permite o processamento flexível em plataformas heterogêneas a partir de protocolos da Web, como HTTP e linguagem de dados com formatos independentes, como JSON para as comunicações entre os agentes. A estrutura do sistema Multiagente é definida a partir de camadas.

Na primeira camada encontram-se os agentes de dispositivos e de recursos, responsáveis pelo gerenciamento de dispositivos (atuadores ou sensores) a partir da execução

de suas operações. A segunda camada é composta pelos agentes coordenadores cujo objetivo é facilitar a comunicação entre os agentes e as demais camadas do sistema. A terceira camada é responsável pela segurança e gerenciamento dos recursos, neste caso estes agentes visam selecionar os serviços necessários para a execução de uma determinada tarefa. Tais informações podem ser obtidas segundo a especificação dos agentes da camada de aplicação (camada mais alta) onde são implementados segundo as características necessárias para cada uma das aplicações.

Apesar de apresentar características de grande complexidade, o trabalho não demonstra como poderia ocorrer a interação entre agentes durante as realizações de tarefas complexas coordenadas. Além disso, o desenvolvimento de aplicações está especificado segundo os comportamentos implementados na camada de aplicação do sistema. Além disso, questões de descoberta e identificação de serviços não são tratados na arquitetura.

Esses trabalhos demonstram o uso de sistemas multiagentes no gerenciamento de dispositivos heterogêneos em ambientes distribuídos. Diversas arquiteturas foram propostas, em sua maioria compostas por camadas. Cada camada tem o objetivo de separar os diversos componentes segundo sua complexidade e funcionalidade.

3.5 Considerações

O estudo sobre os trabalhos apresentados nesta seção demonstra quais são os principais desafios e metodologias utilizados para desenvolver aplicações em ambientes de IoT.

Para permitir o acesso e detecção de dispositivos esses trabalhos adotam tecnologias baseadas em SOA, que garantem funcionalidades *Plug&Play*. Em diversos trabalhos analisados, é utilizada a tecnologia DPWS, que garante a implementação de serviços web em dispositivos. Contudo somente acessar e detectar de dispositivos não são suficientes para atender as necessidades dos usuários em um ambiente da IoT.

Paradigmas como inteligência ambiental e SWOT são focados em aplicações que visam à satisfação das necessidades dos usuários a partir da execução automatizada de tarefas. Em particular a SWOT utiliza-se de tecnologias da Web semântica, como ontologias, de modo a descrever informações contextuais e permitir a anotação semântica de dados acerca dos dispositivos e do ambiente.

Por fim, sistemas multiagentes têm sido utilizados na gestão de dispositivos heterogêneos em ambientes distribuídos, característicos da IoT. Dessa forma, os agentes

podem formar organizações onde cada um dos agentes é responsável por tarefas relacionadas à gestão direta dos dispositivos ou planejamento e supervisão da execução de tais tarefas. Em geral, observa-se que as diversas arquiteturas são organizadas em camadas e definem aos agentes papéis de gerenciadores de recursos ou dispositivos (a nível operacional) e coordenadores (a nível estratégico e de planejamento).

Cada uma das arquiteturas apresenta vantagens e desvantagens apresentadas na Tabela 1, mas observa-se que em geral nenhuma buscou integrar as tecnologias da Web Semântica junto à tecnologia de sistemas multiagentes.

Tabela 1: Comparação entre vantagens e desvantagens das arquiteturas multiagentes estudadas.

Trabalho	Vantagens	Desvantagens
Ângulo-Lopez e Jimenez-Perez (2012)	Integração entre domínios Descrição de conhecimento, descoberta de dispositivos, tomada de decisão, processamento de dados, ações e aspectos de segurança.	Pouca informação sobre a estrutura organizacional e descrição de papéis dos agentes
Pujolle (2006)	Sistema organizado em camadas, objetivando uma melhor distinção entre componentes reativos e cognitivos	Aplicação voltada para tráfego de pacotes em redes.
Fuentes-Fernández et al. (2009)	Integração de dados, gerenciamento de sensores, processamento de dados e adaptação às mudanças	Não há descrição semântica de dispositivos e também não há detecção automática dos dispositivos.
Karnouskos e Tariq (2009)	Sistema organizado em camadas. Uso do protocolo DPWS, permitindo detecção e descrição de dispositivos a nível de serviços.	A definição dos cenários e dados contextuais está codificada nos agentes.
Leong et al. (2014)	Sistema organizado em camadas adotando a arquitetura SOA. Flexibilidade de plataforma e linguagens	Não há detecção automática de dispositivos. Descrição do conhecimento da aplicação definida na camada de aplicação da arquitetura

O uso de ontologias aliado à sistemas multiagentes pode permitir que a arquitetura possa ser adaptativa aos diferentes contextos, bastando para este fim descrever os conceitos associados aos diversos contextos.

Concluindo, a proposta deste trabalho visa integrar características das três tecnologias apresentadas de modo a propor uma arquitetura multiagentes que apresente características *Plug&Play*, possibilite a identificação de recursos e informações contextuais e seja capaz de gerenciar a execução automática de tarefas segundo as necessidades do usuário.

4 ARQUITETURA PARA O GERENCIAMENTO DE DISPOSITIVOS EM AMBIENTES DA IOT

Neste capítulo é apresentada a arquitetura proposta neste trabalho para tratar os aspectos de gerenciamento de dispositivos em ambientes de IoT, destacando as camadas, as funcionalidades bem como a organização de seus componentes. Nas próximas seções são apresentados em detalhes as camadas e os requisitos funcionais da arquitetura proposta, relacionados às características presentes em ambientes de IoT, bem como a especificação de cada um dos componentes.

4.1 Visão geral da arquitetura

Para a concepção da arquitetura (Figura 6) foram considerados alguns aspectos e requisitos importantes para o monitoramento e controle de ambientes de IoT. Esses requisitos foram levantados a partir do estudo da literatura discutido no Capítulo 3.

Os requisitos dos ambientes de IoT envolvem aspectos de interoperabilidade, controle e gestão de dispositivos, e a caracterização de dados contextuais. Para alcançar esses requisitos a arquitetura proposta utiliza as tecnologias de SMA, SOA e recursos da Web Semântica. A arquitetura está organizada em camadas, onde cada uma é responsável por um conjunto de diferentes funções (Figura 6).

As decisões a respeito do número e da composição de cada uma das camadas foram baseadas no estudo da revisão da literatura, onde observa-se em todas as arquiteturas a existência de uma camada de dispositivos. Segundo os trabalhos relacionados à interoperabilidade e em particular a análise do trabalho de Karnouskos e Tariq (2009), os dispositivos são conectados aos agentes do SMA a partir de uma interface de serviços. Cada uma dessas camadas são definidas, não somente segundo a tecnologia empregada, mas também segundo suas especificidades e sua complexidade. A arquitetura é composta pelas seguintes camadas: Camada de Dispositivos, Camada de Serviços e Camada Multiagente, que são detalhadas a seguir.

A camada SMA é subdividida em subcamada cognitiva e subcamada reativa (PUJOLLE, 2006) e os papéis de cada um dos agentes do SMA foram baseados a partir do trabalho de Queiróz (2013), onde é realizado um estudo destacando os papéis mais comuns de SMA's aplicados ao monitoramento e controle de processos industriais.

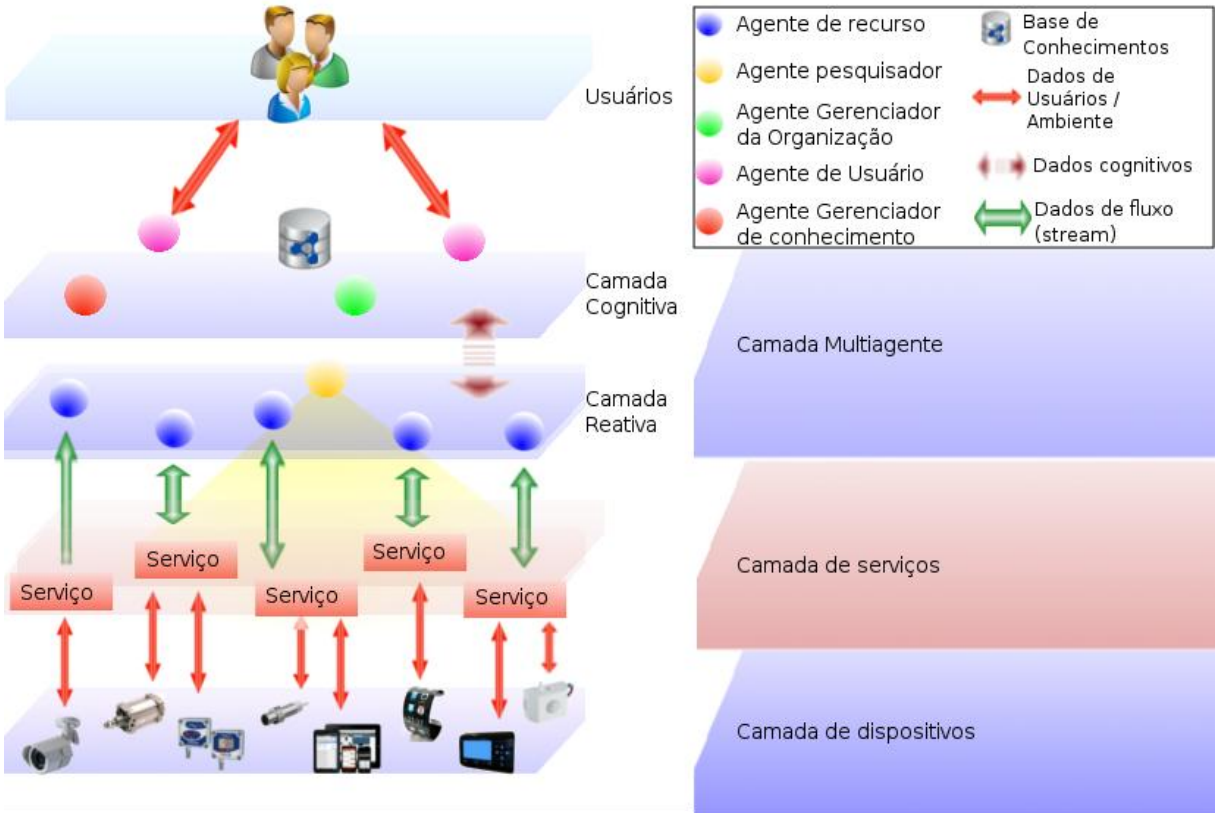

Figura 6: Arquitetura proposta para gerenciamento de dispositivos em ambientes de IoT.

4.1.1 *Camada de dispositivos*

Esta camada é composta pelos dispositivos físicos presentes em ambientes de IoT, correspondendo tanto aos dispositivos instalados no ambiente como também a dispositivos móveis. Estes dispositivos estão conectados à rede e são responsáveis pelo fornecimento de recursos (fornecimento dados ou execução de operações para realização de tarefas), ou servem como interfaces entre os usuários e a infraestrutura da IoT.

Em uma aplicação de IoT, os dispositivos presentes nesta camada correspondem ao ambiente externo, e são conectados ao sistema a partir da Camada de Serviços. Isso significa que uma importante característica presente nestes dispositivos é a capacidade de se conectar ao sistema através de interfaces padronizadas baseadas em serviços (SOA).

4.1.2 *Camada de serviços*

Esta camada fornece uma interface de comunicação comum para os dispositivos. Para isso é proposto a utilização de um *middleware* de serviços, responsável por encapsular os

dispositivos e prover o acesso a seus recursos e suas funcionalidades por meio de uma interface padronizada. Além disso, essas interfaces devem atender às especificações de SOA para a publicação, descoberta e acesso desses serviços na rede.

Para atender os requisitos dos dispositivos presentes em ambientes de IoT, tecnologias utilizadas nesta camada da arquitetura devem suportar o desenvolvimento de infraestruturas baseadas em serviços para dispositivos com restrições de recursos (memória e processamento). Além disso, devem permitir a identificação e a descrição os recursos, serviços e operações, que cada dispositivo apresenta, garantindo que o sistema obtenha as funcionalidades de qualquer novo dispositivo conectado. E por fim, suportar funcionalidades de *Plug&Play* que contribuem para a flexibilidade das aplicações.

4.1.3 *Camada multiagente*

Esta camada é a mais complexa, sendo responsável pela integração e a gestão dos recursos. A camada multiagente é constituída por um conjunto de agentes responsáveis por desempenhar diferentes funções. Essas funções estão distribuídas e agrupadas em um conjunto de papéis e definem atividades específicas que cada agente desempenha no sistema. Conceitualmente, esta camada pode ser dividida em duas novas subcamadas denominadas de Camada Reativa e Camada Cognitiva (PUJOLLE, 2006) (Figura 6).

A Camada Reativa compreende os agentes responsáveis por tratar diretamente cada recurso conectado ao sistema. As ações dos agentes que compõem esta camada são mais rápidas, envolvendo o mecanismo de estímulo e resposta. Isso se deve ao fato dos dispositivos monitorados por estes agentes necessitarem tempos de respostas muito baixos.

A Camada Cognitiva é responsável pelas tomadas de decisões mais complexas e a execução de planos num nível mais estratégico. Desse modo, os agentes que compõem esta camada atuam a partir de um tempo de resposta mais longo, envolvendo atividades de planejamento, controle e supervisão. Além dos agentes, esta camada é composta por uma Base de Conhecimentos estruturada em ontologias e será descrita em detalhes na seção 4.3.

Uma visão geral das duas subcamadas é dada a seguir.

- **Subcamada Reativa**

Esta camada é formada por agentes reativos que estão encarregados da detecção, comunicação e a gestão dos dispositivos físicos. Estes agentes acessam os dispositivos a partir

da Camada de Serviços. Dessa forma, podem fazer a requisição das informações ou solicitar a execução de operações disponibilizadas por estes dispositivos.

Nesta camada os agentes desempenham o papel de Agente de Recursos ou Agente Pesquisador. Esses papéis, assim como os outros definidos na arquitetura proposta são especificados na Seção 4.2.1.

- **Subcamada Cognitiva**

Os agentes que compõem esta camada são responsáveis pela integração e gerenciamento dos recursos e pela execução das funcionalidades da aplicação. Para isso, esses agentes utilizam as informações e os dados disponibilizados pelos Agentes de Recurso, além do conhecimento disponível sobre os recursos da aplicação presentes na Base de Conhecimentos. Outra função desempenhada pelos agentes desta camada consiste em determinar os contextos da aplicação, relacionar os hábitos do usuário com uma dada tarefa, gerenciar os Agentes de Recurso na execução de tarefas. Para execução deste conjunto de funções, os agentes dessa camada assumem três papéis, especializados na execução de determinadas tarefas. Os papéis dos agentes desta camada são denominados: Agente Gerenciador de Conhecimento, Agente Gerenciador de Organização e Agente de Usuário.

4.2 Organização social do SMA

Nessa seção são detalhados os papéis desempenhados pelos agentes na Camada Multiagente (Figura 7). Em aplicações locais, onde a arquitetura está em execução e os dispositivos que fornecem os recursos se encontram em uma rede local (LAN), é necessário somente um único Agente Pesquisador. Contudo, em aplicações distribuídas, pode existir mais de um agente desempenhando este papel, onde cada um é responsável por monitorar a entrada dos dispositivos em sua rede. Os demais papéis (Agente Gerenciador de Conhecimento e o Agente Gerenciador de Organização) podem ser desempenhados por um único agente. Contudo, o Agente Gerenciador de Organização pode criar diferentes organizações para cada tipo de contexto. Cada um dos papéis são descritos nas próximas seções.

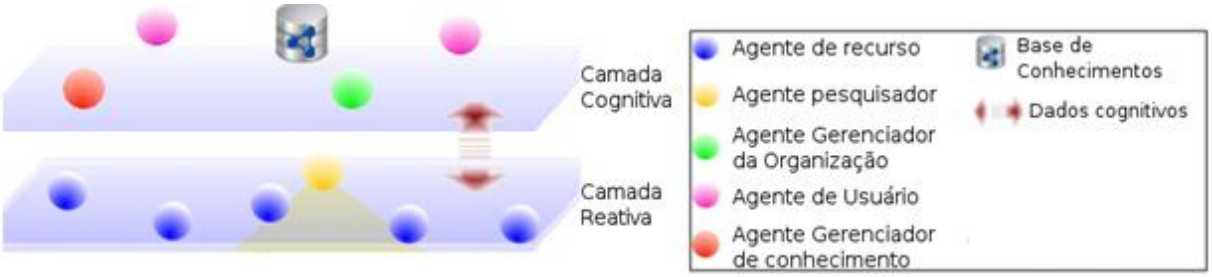

Figura 7: Papéis dos agentes da camada Multiagentes da arquitetura proposta para gerenciamento de dispositivos em ambientes de IoT.

4.2.1 Descrição dos papéis dos agentes

Na arquitetura proposta os agentes são definidos em termos dos papéis desempenhados. Os papéis dos agentes definem os comportamentos que os mesmos devem apresentar na aplicação, por meio da definição das tarefas que estes desempenham, bem como a interação entre eles. Em uma aplicação podem existir um ou mais agentes desempenhando o mesmo papel, embora encarregados de executar tarefas especializadas. Ao mesmo tempo um único agente pode desempenhar vários papéis, isto significa que este agente irá implementar os diferentes comportamentos definidos nesses papéis. Nesse trabalho os comportamentos dos agentes são especificados com base no modelo de agentes BDI, que é definido em função das crenças, planos e objetivos que os agentes devem apresentar.

4.2.1.1 Agente de Recursos

Esse papel define um conjunto de atividades voltadas para o gerenciamento de dispositivos de acordo com seus requisitos e especificações. Além do gerenciamento de dispositivos, os agentes que assumirem este papel devem implementar funcionalidades para realizar o monitoramento e a análise do fluxo de dados disponibilizados pelo dispositivo. Os agentes que desempenham este papel também devem monitorar o estado das variáveis dos dispositivos, informando aos agentes da Camada Cognitiva eventuais mudanças e anormalidades.

Dependendo do tipo de dispositivo, este agente pode ter como responsabilidade o controle do estado de uma ou mais variáveis. Este controle pode ser obtido a partir de duas diferentes maneiras: o controle das variáveis que determinam o estado do ambiente, geralmente é realizado por atuadores e sensores; e, o controle do estado interno dos

dispositivos, como por exemplo, consumo de energia, utilização de recursos de processamento entre outros.

Nas tarefas de monitoramento e análise, os resultados devem ser transformados em dados simbólicos dotados de semântica. Para a realização dessas tarefas, os agentes que desempenham esse papel devem analisar os dados obtidos segundo o conhecimento disponível na ontologia e informar os dados anotados semanticamente, segundo um período de tempo, o estado das variáveis monitoradas. Esses valores representam o conhecimento atual sobre as condições do ambiente, que foram medidas pelos elementos sensores dos dispositivos, as quais são informadas aos agentes da Camada Cognitiva para o gerenciamento da aplicação.

A Figura 8 resume os comportamentos definidos por esse papel. O diagrama foi baseado na figura de uma classe dos diagramas de classe da UML. Entretanto, os atributos representam as crenças (o conhecimento) dos agentes, enquanto os métodos representam as tarefas que os mesmos devem prover.

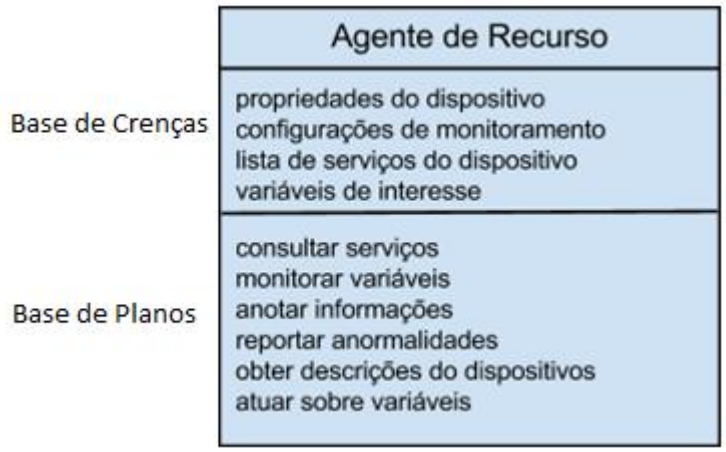

Figura 8: Comportamentos do Papel dos Agentes de Recurso.

As funções executadas pelo papel de Agente de Recurso demandam como requisito a comunicação com os dispositivos de modo a acessar tais informações e recursos. Isso é realizado pelos serviços disponibilizados pela Camada de Serviços.

4.2.1.2 Agente Pesquisador

O agente que desempenhar esse papel é responsável pelo acompanhamento e a procura contínua de dispositivos na Camada de Serviços. Além disso, é encarregado da inicialização

de Agentes de Recursos para gerenciar os dispositivos e recursos descobertos na rede. Assim, quando um novo dispositivo for encontrado ou conectado, os agentes que executam este papel são responsáveis por recuperar os metadados desses dispositivos e posteriormente com o auxílio de um Agente Gerenciador de Conhecimento, procurar na Base de Conhecimentos a descrição semântica associada ao dispositivo e seus recursos.

A descrição semântica dos diversos tipos de dispositivos é definida previamente e armazenada na Base de Conhecimentos da aplicação. Esta descrição permite associar semântica aos dispositivos e as informações coletadas. Dessa forma, este agente passa a ter seus recursos descritos semanticamente, os quais podem ser encontrados e usados pelos outros agentes para o monitoramento e controle do ambiente.

Na Figura 9 é apresentado um diagrama que ilustra os comportamentos desse papel.

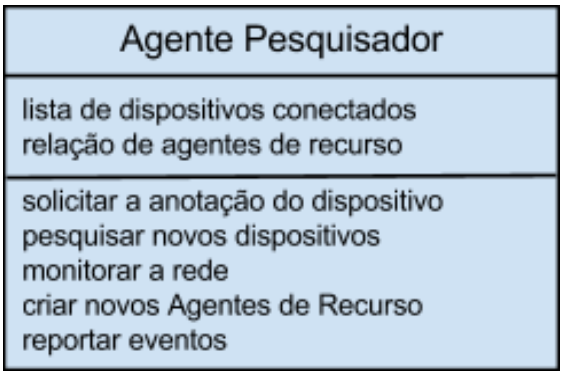

Figura 9: Comportamentos do Papel do Agente Pesquisador.

4.2.1.3 Agente Gerenciador de Conhecimento

O agente que desempenhar o papel de Gerenciador de Conhecimento é responsável por gerenciar todo o conhecimento da aplicação que inclui a descrição semântica dos modelos de dados, dispositivos e processos. Esta descrição é armazenada na Base de Conhecimento e estruturada de acordo com as ontologias da aplicação. Dessa forma, as principais funções desse agente envolvem prover uma interface de acesso ao conhecimento da aplicação para os outros agentes do SMA, permitindo a consulta, a alteração e a inserção de novos conhecimentos.

Na Figura 10 é apresentado um diagrama que ilustra os principais comportamentos definidos por esse papel.

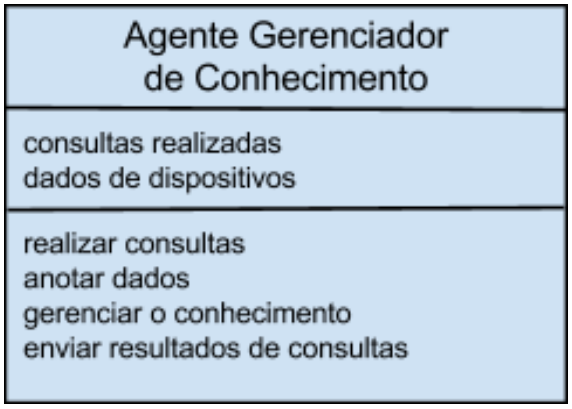

Figura 10: Comportamentos do Papel do Agente Gerenciador de Conhecimento.

4.2.1.4 Agente Gerenciador de Organização

Os agentes que assumirem o papel de Gerenciador de Organização são responsáveis por determinar contextos e criar as organizações para a realização das tarefas associadas aos contextos identificados. Determinar o contexto da aplicação significa analisar os recursos presentes (dispositivos e informações) e associá-los com as tarefas que podem ser realizadas a partir dos mesmos. Os contextos são descritos nas ontologias da aplicação e ficam armazenados na base de conhecimento. Quando é identificada a existência de um contexto, esse agente estabelece uma nova organização, formada por Agentes de Recurso que estão aptos para a realização das tarefas correspondentes a este contexto. Da mesma forma, quando ocorrem mudanças no contexto, esse agente é responsável por tomar as medidas necessárias, como por exemplo, abortar os planos dessa organização.

Este agente pode criar organizações de uma forma dinâmica, definindo o conjunto dos objetivos a serem alcançados por cada um dos Agentes de Recurso presentes no sistema. Na aplicação, podem haver vários diferentes contextos. Dessa forma, também podem existir diferentes organizações de Agentes de Recurso.

Na Figura 11 é apresentado o diagrama que ilustra os comportamentos definidos por esse papel.

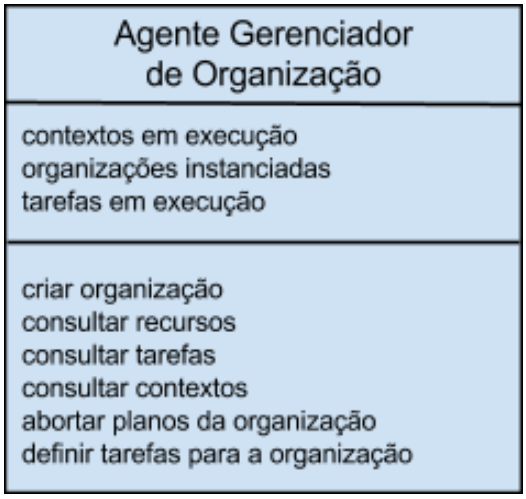

Figura 11: Comportamentos do Papel do Agente Gerenciador de Organização.

4.2.1.5 Agente de Usuário

Os agentes com o papel Agente de Usuário são responsáveis por fazerem a interface entre os usuários e a aplicação, apresentando as funcionalidades da mesma através de interfaces gráficas ou atendendo suas requisições. Entre as responsabilidades desse agente estão: gerenciar as requisições de usuários e verificar a execução de tarefas relacionadas a estes perfis. De acordo com a aplicação, podem existir vários agentes desempenhando o papel de Agente de Usuário, sendo que cada um é responsável por um usuário em específico. Além das ações anteriormente descritas, o Agente de Usuário também pode influenciar na identificação de quais contextos são aplicáveis a um dado usuário, fornecendo informações do perfil usuário para o Agente Gerenciador de Organização.

Na Figura 12 é apresentado o diagrama que ilustra os comportamentos definidos por esse papel.

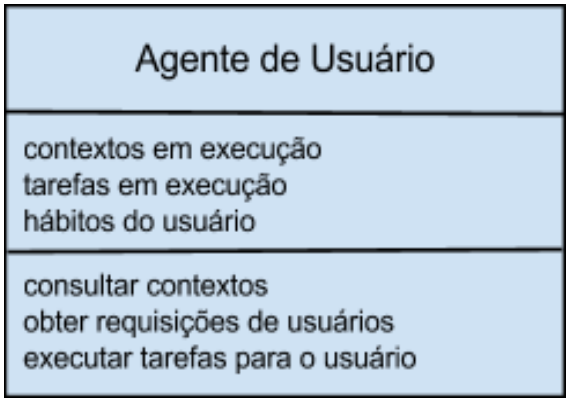

Figura 12: Comportamentos do Papel do Agente de Usuário.

4.2.2. Interação entre os agentes

Em uma aplicação os agentes interagem para troca de recursos e prestação de serviços. Nesse sentido, nessa seção são apresentados os tipos de interação existentes entre os agentes em função dos papéis desempenhados durante a execução de alguns cenários.

4.2.2.1 Inicialização do SMA

Este cenário representa as ações executadas pelos agentes no momento que o sistema é inicializado para desempenhar suas funções (Figura 13). Na inicialização, é necessário iniciar a execução dos agentes que desempenham os papéis Agente Pesquisador, Agente Gerenciador de Conhecimento e o Agente Gerenciador de Organização. Estes agentes são encarregados de preparar e inicializar o ambiente da aplicação.

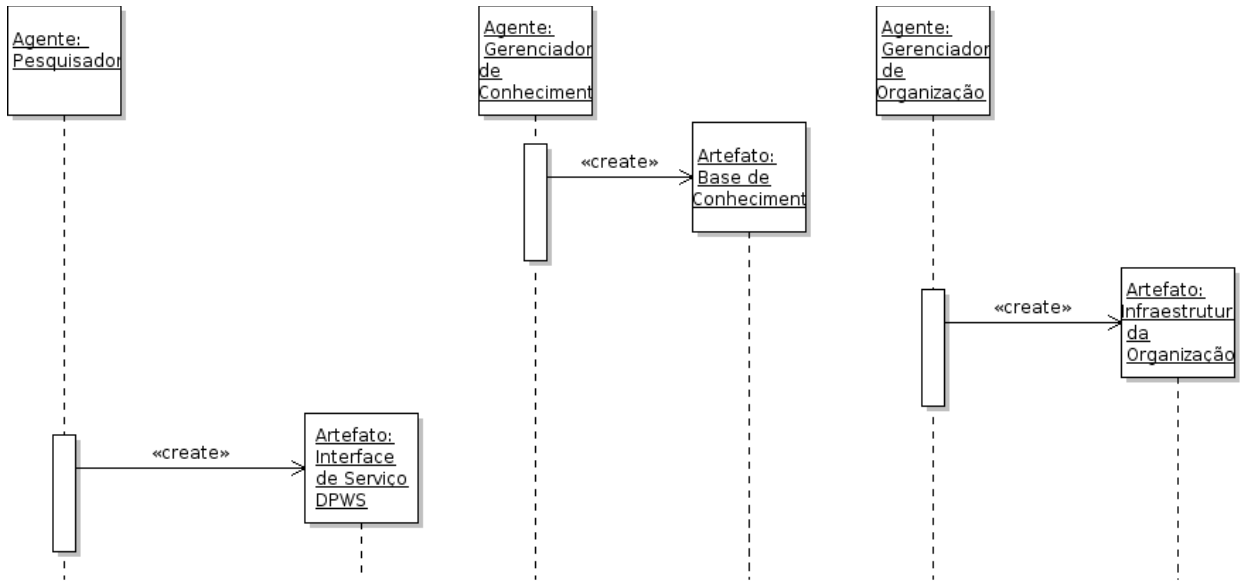

Figura 13: Inicialização do SMA.

O agente desempenhando o papel de Agente Gerenciador do Conhecimento deve estabelecer a conexão com a Base de Conhecimentos. O agente que desempenha o papel de Agente Pesquisador é responsável pela inicialização das interfaces da Camada de Serviço.

Isso significa que este agente deve inicializar o processo de monitoramento da rede para detectar os dispositivos conectados.

Por fim, o Agente Gerenciador de Organização, iniciará a infraestrutura de controle da organização de agentes.

4.2.2.2 Descoberta de dispositivos e descrição semântica

Neste cenário participam os agentes desempenhando o papel de Agente Pesquisador, Agente Gerenciador de Conhecimento e Agente de Recurso. Esses agentes interagem com o objetivo de realizar as tarefas para a detecção de dispositivos e a posterior busca pelas informações semânticas.

A descoberta de dispositivos e busca pelas informações semânticas (Figura 14) são realizadas sempre que um novo dispositivo é conectado à(s) rede(s) que está(ão) sendo monitorada(s). O Agente Pesquisador monitora de forma continua a rede à procura de novos dispositivos conectados ou dos previamente conectados. Uma vez que o dispositivo foi identificado e foi obtida a sua descrição, o Agente Pesquisador solicita ao Agente Gerenciador de Conhecimento que encontre as informações semânticas do dispositivo. Utilizando as informações semânticas, o Agente Pesquisador irá inicializar um Agente de Recurso para gerenciar este novo dispositivo.

A troca de mensagens entre estes agentes (Agente de Recurso e Agente Pesquisador), depois de obtidos os dados descritivos do dispositivo, tem como remetente o Agente Pesquisador e como destinatário o Agente de Recursos recém instanciado. Em seu conteúdo encontram-se, além das informações semânticas, o nome do dispositivo, *namespace* e *endpoint*, a partir dos quais o Agente de Recurso poderá acessá-lo.

O Agente de Recurso agora buscará no dispositivo a descrição de todos os serviços e operações. Em seguida, o Agente de Recursos solicita ao Agente Gerenciador de Conhecimento a informação semântica dos serviços disponíveis no dispositivo. Os procedimentos acima ocorrerão para todos os novos dispositivos conectados ao sistema.

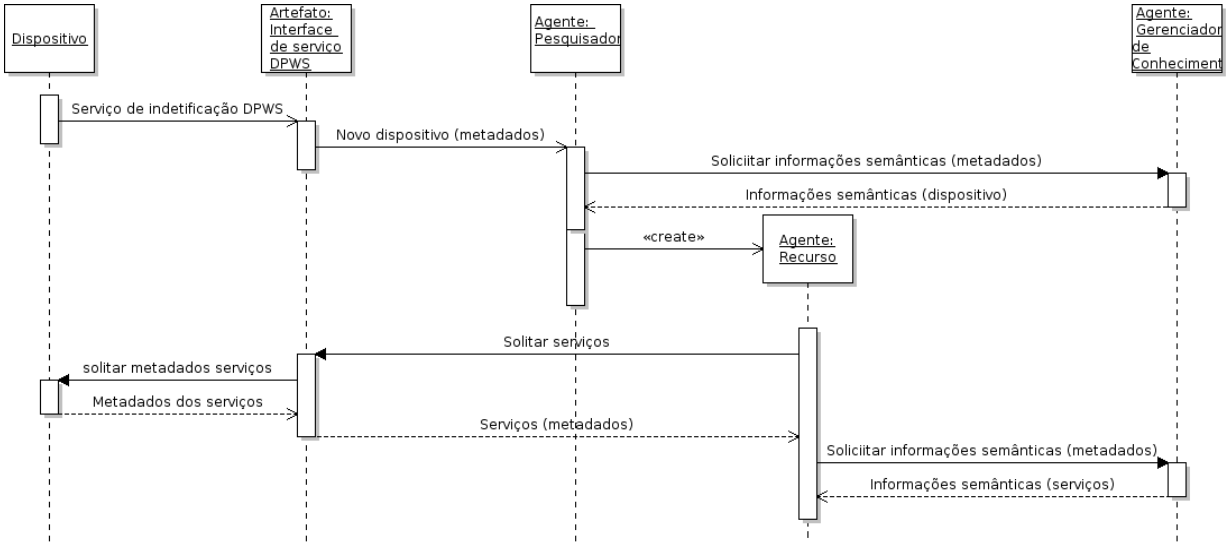

Figura 14: Descoberta de dispositivos e descrição semântica.

4.2.2.3 Determinação do contexto

A determinação do contexto é realizada pelo Agente Gerenciador da Organização, depois de realizada a busca pelas informações semânticas dos dispositivos e de seus serviços. Os papéis envolvidos neste cenário são: o Agente de Recursos, o Agente Gerenciador de Organização e o Agente Gerenciador de Conhecimento.

A execução deste cenário (Figura 15) é iniciada com os Agentes de Recurso informando para o Agente Gerenciador de Organização os recursos gerenciados por ele. Com as informações sobre os dispositivos presentes no ambiente e seus recursos, o Agente Gerenciador de Organização solicita ao Agente Gerenciador de Conhecimento se existe algum contexto associado aos recursos disponíveis.

Uma vez determinado o contexto da aplicação, o Agente Gerenciador de Conhecimento informa ao Agente Gerenciador de Organização as tarefas que poderão ser realizadas de acordo com o contexto. Isso significa que o Agente Gerenciador de Organização pode criar uma nova organização formada por Agentes de Recurso para gerenciar os dispositivos na realização das tarefas associadas ao contexto.

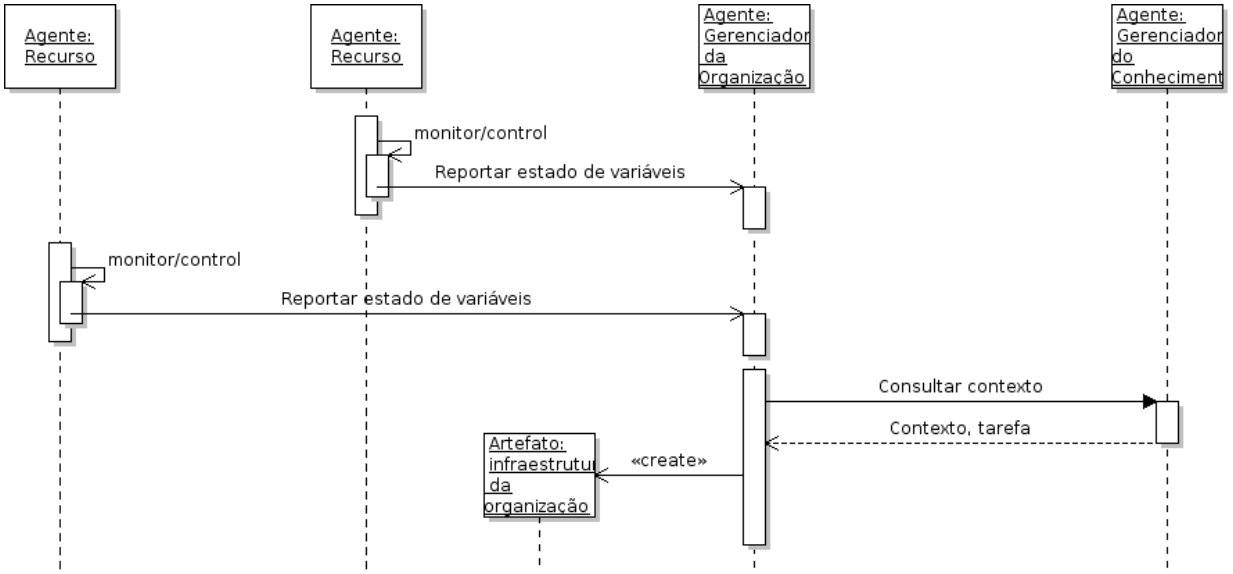

Figura 15: Determinação do contexto da aplicação.

As mensagens trocadas entre os Agentes de Recursos e o Agente Gerenciador de Organização apresentam como conteúdo os estados das variáveis monitoradas ou alertas de anormalidades durante a execução da tarefa. Esta mensagem tem como objetivo verificar se o contexto ainda é válido para aquela tarefa.

4.2.2.4 Gerenciamento de dispositivos

O gerenciamento de dispositivos (Figura 16) ocorre sempre que uma nova organização é criada para tratar determinado contexto. Neste caso, o Agente Gerenciador de Organização tem a responsabilidade de criar a organização para coordenar os Agentes de Recurso na realização das tarefas.

Uma vez estabelecida a organização, o Agente Gerenciador de Organização atribui as tarefas aos Agentes de Recurso para monitorar ou controlar as condições do ambiente, de acordo com as normas determinadas pela organização.

A execução dos planos dos diferentes agentes envolvidos na organização permite ao sistema tratar a execução de tarefas de maior complexidade e diferentes contextos dinamicamente.

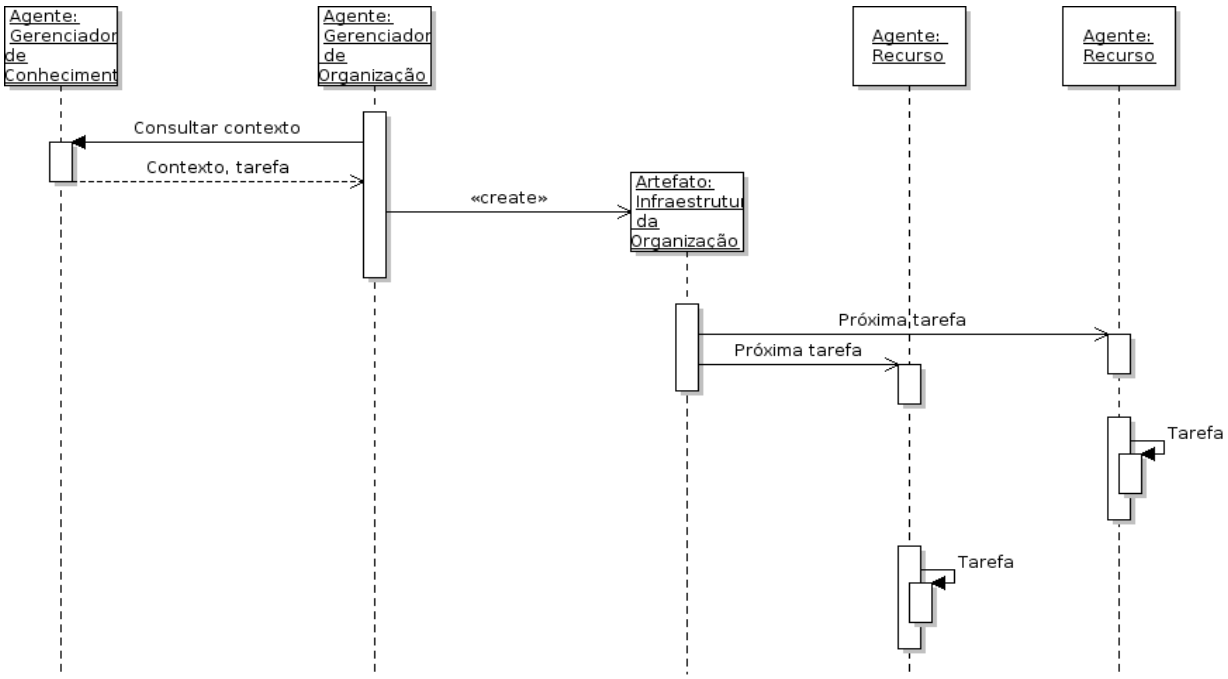

Figura 16: Gerenciamento de dispositivos.

4.2.2.5 Determinação de dispositivos de interface de usuário

A determinação de dispositivos de interface de usuário ocorre quando um Agente Gerenciador de Organização, após consultar os serviços consegue identificar um dispositivo que permite aos usuários acessarem e utilizarem as funcionalidades oferecidas pela aplicação. Essas funcionalidades abrangem desde a configuração da aplicação até informações sobre as condições e tarefas que foram ou estão sendo executadas.

A execução deste cenário (Figura 17) ocorre depois que um Agente de Recursos informa os serviços disponíveis no dispositivo ao Agente Gerenciador de Organização. Ao receber estes dados o Agente Gerenciador de Organização consulta através do Agente Gerenciador de Conhecimento a descrição do dispositivo.

Ao receber o resultado da consulta, o Agente Gerenciador de Organização, pode identificar a partir dos serviços e da descrição do dispositivo, aqueles recursos que permitem a interação com usuários (dispositivos de interface homem-máquina, como dispositivos móveis). Neste caso é gerado a instancia de um Agente de Usuário para gerenciar o Agente de Recurso associado àquele dispositivo.

O Agente de Usuário, segundo as necessidades do usuário, vai solicitar aos Agentes de Recursos as informações sobre as variáveis e estados associados.

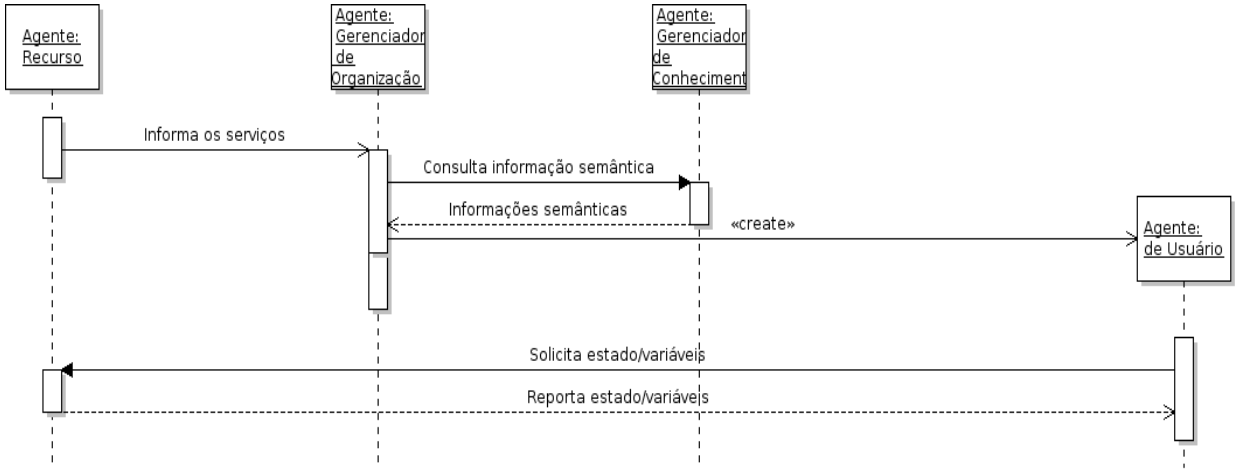

Figura 17: Determinação de dispositivos de interface de usuário.

4.2.2.6 Interação do usuário com o sistema

A execução da requisição de usuário ocorre quando um usuário do sistema deseja que os dispositivos presentes no ambiente executem alguma tarefa específica (Figura 18). Os agentes que participam deste cenário são o Agente Gerenciador de Organização, o Agente Gerenciador de Conhecimento e os Agentes de Usuário.

O cenário começa quando um usuário, usando um dispositivo de interface com o sistema, realiza a requisição de uma dada tarefa. O Agente de Usuário recebe esta solicitação e a repassa para o Agente Gerenciador de Organização que iniciará a execução da tarefa.

Durante a realização da tarefa os dados relacionados ao estado de tarefas e das variáveis de interesse serão exibidas para o usuário a partir do dispositivo de interface.

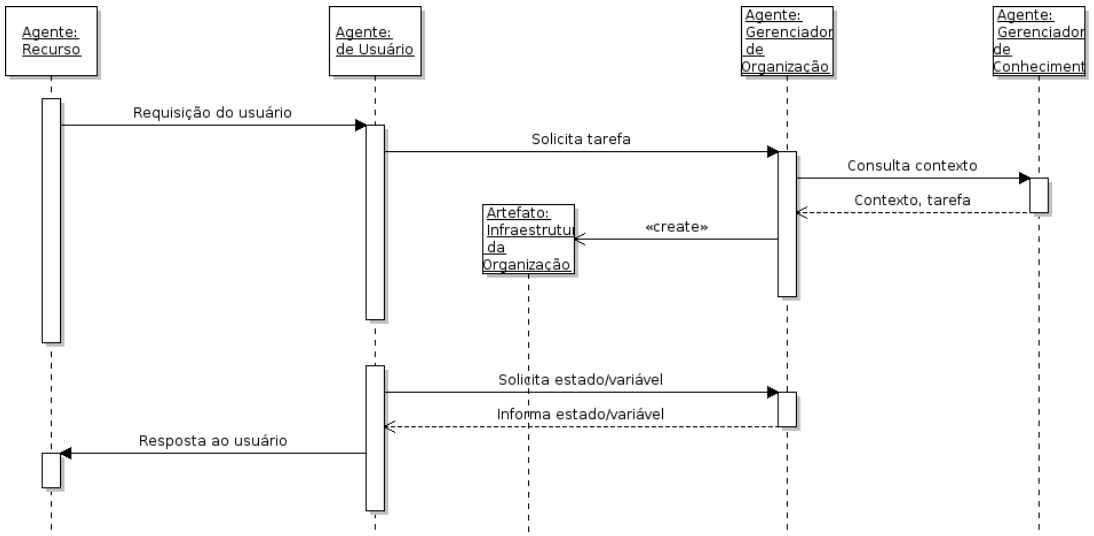

Figura 18: Execução de requisições de usuário.

4.3 Base de Conhecimentos

A Base de Conhecimento da aplicação consiste de um repositório semântico que contém um conjunto de ontologias. Essas ontologias são utilizadas para descrever todo o conhecimento envolvido no domínio da aplicação. Elas podem ser desenvolvidas utilizando *frameworks* e linguagens padronizadas como RDF/OWL, que são definidas pela W3C e amplamente aceitas pela comunidade. Além do desenvolvimento das ontologias a serem utilizadas pela aplicação também podem ser reutilizadas ontologias já desenvolvidas, desde que essas atendam aos requisitos da aplicação.

Nesse sentido, os agentes da Camada Cognitiva utilizam o conhecimento definido na Base de Conhecimento para a coordenação e controle dos outros agentes, bem como para o gerenciamento e planejamento das ações.

Na Base de Conhecimento do sistema estão descritos os conceitos relacionados aos dispositivos, às tarefas, aos contextos, às organizações dos agentes e aos perfis de usuários.

Para atender os propósitos da arquitetura proposta foi desenvolvida uma ontologia de aplicação.. Na Figura 19 é apresentada uma representação visual da estrutura da ontologia criada em termos dos principais conceitos e seus relacionamentos.

A ontologia relaciona três conceitos fundamentais (ou principais) denominados: Dispositivo, Tarefa e Contexto. Os dispositivos (Dispositivo) provém serviços (Serviço), que são constituídos de operações (Operações), que por sua vez apresentam parâmetros (Parâmetros) que podem ser simples ou complexos

O conceito Tarefa está relacionado à Dispositivo e Contexto. Isso significa que toda tarefa realizada pelo sistema depende dos dispositivos disponíveis e de um contexto, definindo as ações relacionadas.

O contexto é descrito por conceitos necessários para determinar situações, eventos ou entidades. Dessa forma, o Contexto pode ser um contexto de tempo (ContextoDeTempo), permitindo a identificação de eventos relacionados a tempos como horários, períodos, entre outros ou contexto de identidade (ContextoDeIdentidade), permitindo a identificação de recursos e entidades.

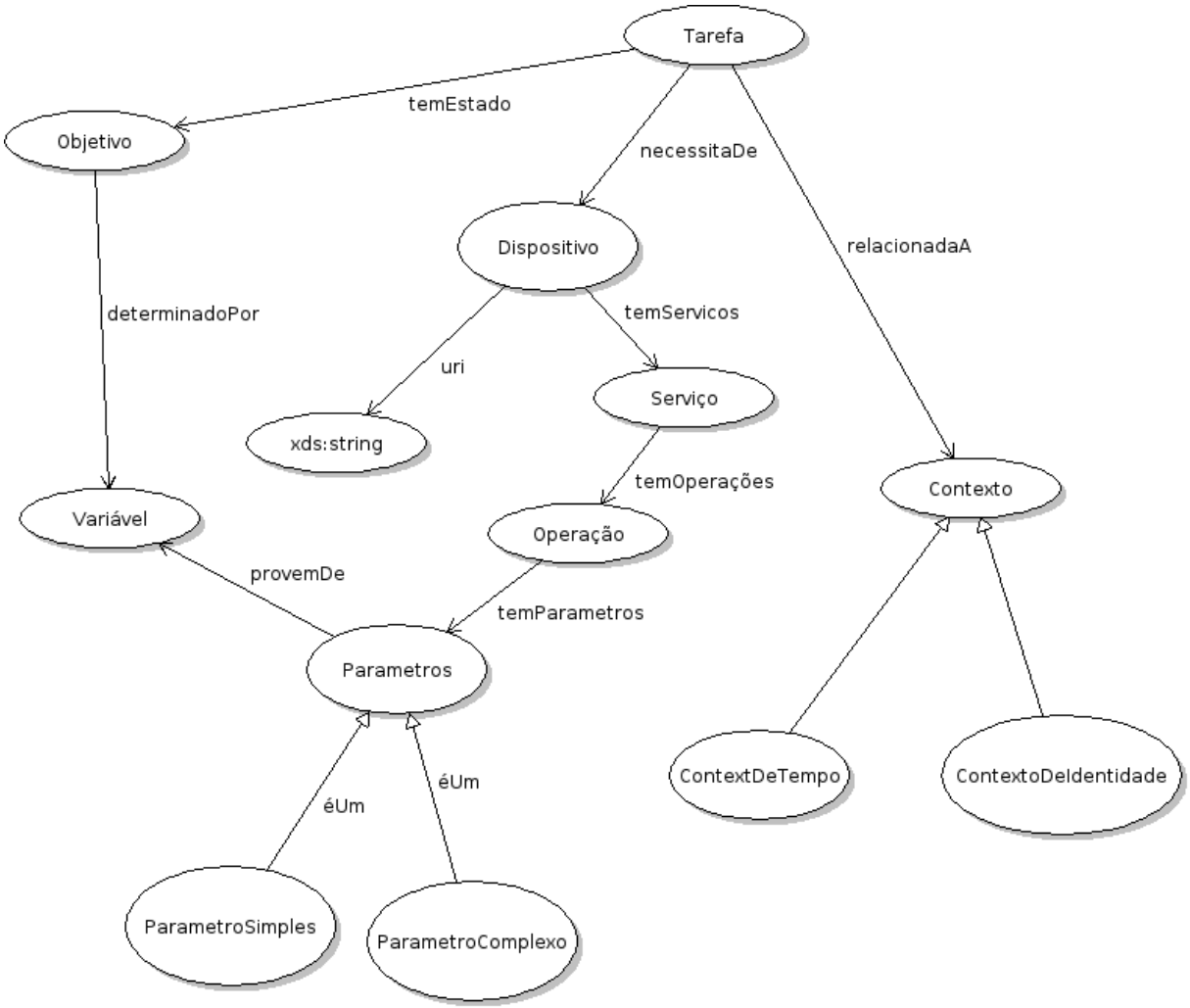

Figura 19: Ontologia de aplicação do sistema.

5. IMPLEMENTAÇÃO DA ARQUITETURA

Este capítulo apresenta a implementação da arquitetura proposta. Neste caso, serão detalhadas quais foram as tecnologias utilizadas na implementação de cada uma de suas camadas e como ocorreu a integração entre estas diferentes tecnologias.

5.1 Visão geral da Implementação da Arquitetura

Inicialmente para a implementação da arquitetura um conjunto de tecnologias (apresentadas no Capitulo 2) foram utilizadas. Assim, os componentes de cada uma das camadas foram desenvolvidos a partir de tecnologias específicas. Na camada de serviços foi utilizado o *framework* JMEDS[8] para implementar os serviços baseados no protocolo DPWS. Para implementar todo o SMA foi utilizado o *framework* JaCaMo. O desenvolvimento das ontologias foi realizado com a ferramenta Protégé[9]. A ferramenta Jena[10] foi utilizada para a manipulação do conhecimento usado pela aplicação e estruturado segundo as ontologias. As tecnologias envolvidas na implementação da arquitetura são ilustradas na figura 20.

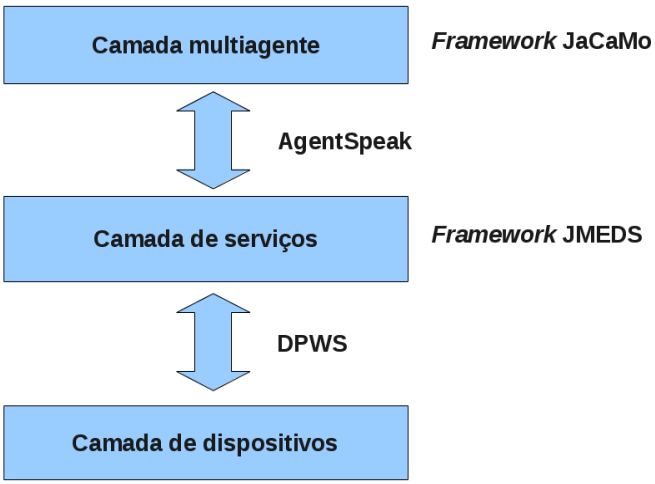

Figura 20: Tecnologias envolvidas na implementação da arquitetura proposta.

[8] http://sourceforge.net/projects/ws4d-javame/

[9] http://protege.stanford.edu/

[10] https://jena.apache.org/

5.2 Implementação da Camada de Serviços

A integração entre a Camada de dispositivos e a Camada multiagente é feita através da implementação de uma interface de comunicação comum baseada em serviços. Essa camada segue os princípios de uma Arquitetura Orientada a Serviços, sendo representada, basicamente, por dois tipos de componentes: os componentes que representam os provedores de serviços (os quais proveem serviços e recursos) e aqueles que representam os clientes (os quais consomem os serviços e recursos dos provedores). Nesse sentido, o *framework* JMEDS é usado para a implementação tanto do cliente como dos provedores de serviço.

O *framework* JMEDS, por se tratar de uma implementação leve e enxuta de um protocolo de serviços (DPWS) permite que o próprio dispositivo execute o código e funcione como um provedor de recursos para os clientes em uma rede local. Os clientes dessa camada consistem em componentes de *software*, implementados utilizando a API fornecida pelo JMEDS, responsáveis por fazer a interface com os agentes da Camada Multiagente. Os provedores de serviços, por sua vez, utilizam os componentes que fazem uma interface direta com os dispositivos na Camada de Dispositivos,

Com base nessa tecnologia é possível encapsular as funcionalidades dos dispositivos como serviços DPWS, além disso, a tecnologia DPWS, suporta funcionalidades que permitem a descoberta e o consumo desses serviços. A classe “DPWSClient” implementa todas as funcionalidades do cliente DPWS (Figura 21).

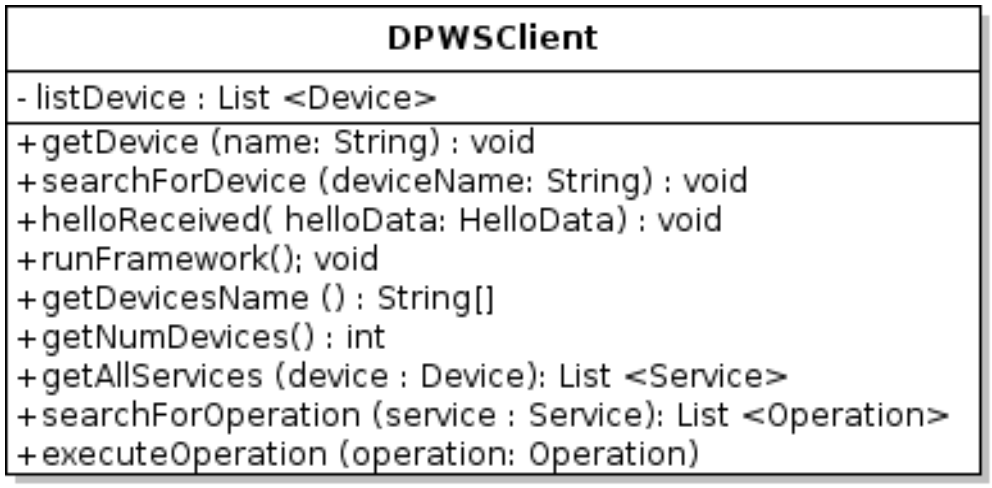

Figura 21: Diagrama de classe do cliente da camada de serviços.

A classe "DPWSClient" é formada pela composição da classe "Device", presente na API. A classe "Device" é instanciada durante a execução do método "helloReceived" que é

executado de maneira assíncrona no momento que um novo dispositivo é conectado à rede. No DPWS, existem três elementos básicos: dispositivo, serviço e operações. Todo dispositivo apresenta serviços e todo serviço apresenta operações. Dessa forma, os demais métodos são relacionados à pesquisa, consulta e execução de cada um desses elementos.

Para o estudo de caso, descrito no capítulo 6, não foram utilizados dispositivos reais implementando serviços DPWS. Neste caso, os dispositivos foram simulados usando recursos fornecidos pelo *framework* JMEDS. A Figura 22 mostra o diagrama de classe dos componentes implementados como os provedores de serviços simulando os dispositivos reais.

A classe "ServiceProviders" tem a função de inicializar os diferentes dispositivos, representados pela classe "Device". Neste caso, cada dispositivo é uma *thread* conectada a um IP e uma porta independente, representando um conjunto de dispositivos conectados a uma mesma rede. Cada dispositivo apresenta um conjunto de serviços (classe "Service"), que por sua vez apresenta um conjunto de operações (classe "Operation"). As operações representam as ações que cada dispositivos pode realizar sendo acessadas a partir de um serviço específico.

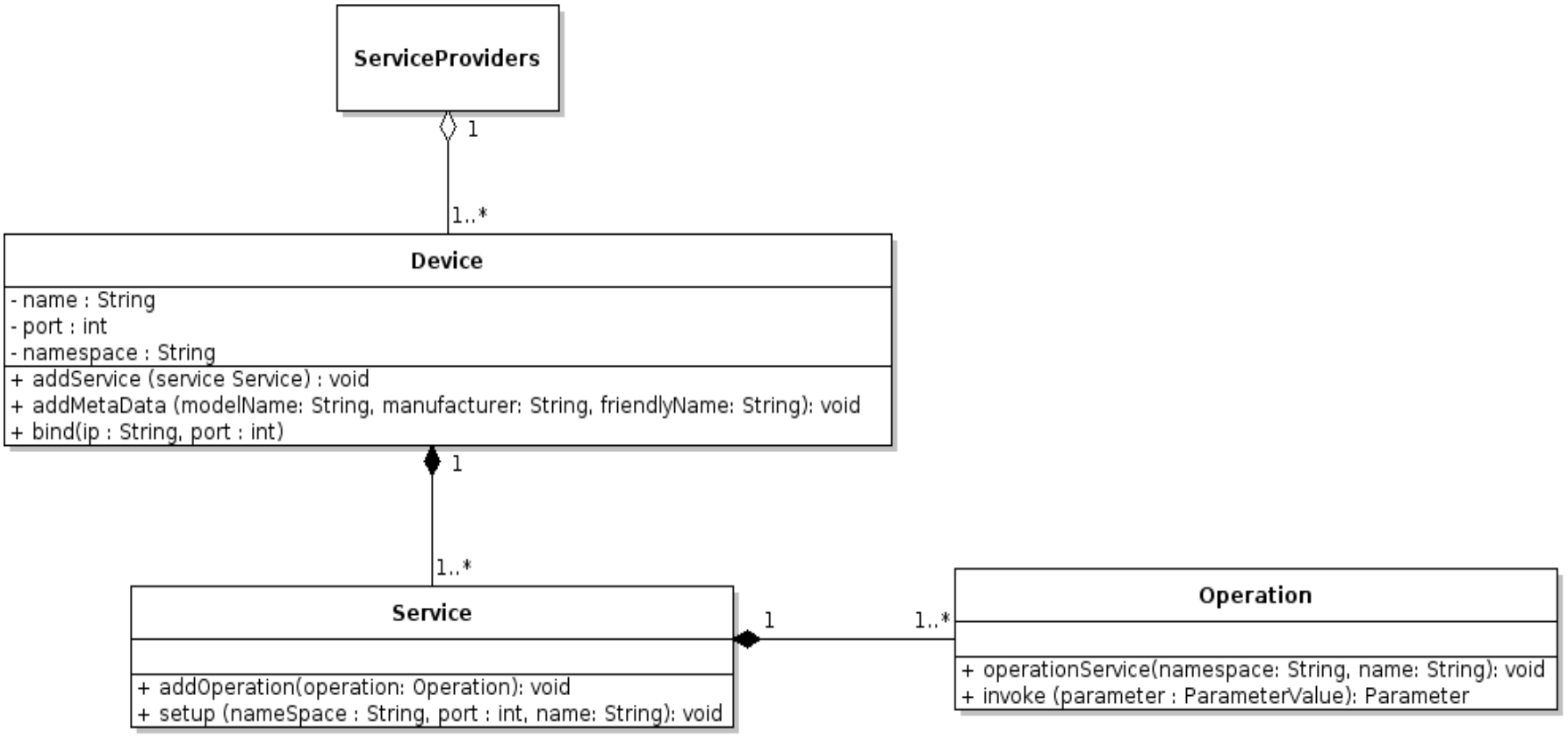

Figura 22: Diagrama de classes dos dispositivos simulados.

5.3. Implementação da Camada Multiagentes

Na Camada Multiagentes, os agentes foram implementados utilizando o *framework* JaCaMo (Jason, CArtAgO e Moise), que além de fornecer APIs para o desenvolvimento do código dos agentes, também fornece uma plataforma com toda a infraestrutura necessária para os testes e a execução dos agentes.

O *framework* Jason é utilizado para implementar cada um dos agentes presentes no sistema. O Jason também fornece a infraestrutura para o gerenciamento do ciclo de vida dos agentes, permitindo em tempo de execução a criação de novos agentes ou a exclusão dos existentes. Além disso, permite a implementação dos agentes a partir da linguagem *AgentSpeak*, linguagem do paradigma lógico que permite uma sintaxe mais intuitiva. Por fim, Jason também permite a criação de novos papéis de agentes, sem a necessidade de parada do sistema.

O *framework* CArtAgO foi utilizado para permitir com que os agentes implementados a partir do *framework* Jason possam acessar as funcionalidades das tecnologias utilizadas na Camada de Serviços e na Base de Conhecimentos.

Para o fornecimento das funcionalidades DPWS, duas classes de artefatos são utilizadas, onde cada uma apresenta funcionalidades específicas, necessárias de acordo com o papel interpretado por cada um dos agentes. Estas classes são denominadas “Device” e “SearchingBoard". A classe denominada “Device”, é uma classe Java que implementa o artefato utilizado pelos Agentes de Recurso. Esta classe representa um cliente DPWS permitindo ao Agente de Recurso consultar, recuperar e executar os serviços e as operações disponibilizadas pelo dispositivo, assim como enviar e receber parâmetros a partir da execução das operações especificadas pelo dispositivo.

A classe “SearchingBoard” também representa um cliente DPWS, o qual serve de suporte para as funcionalidades do papel Agente Pesquisador, onde são implementadas funções de pesquisa e sinais relacionados à descoberta de novos dispositivos na rede. Ambas as classes fazem interface com a classe "DPWSClient" (Figura 23).

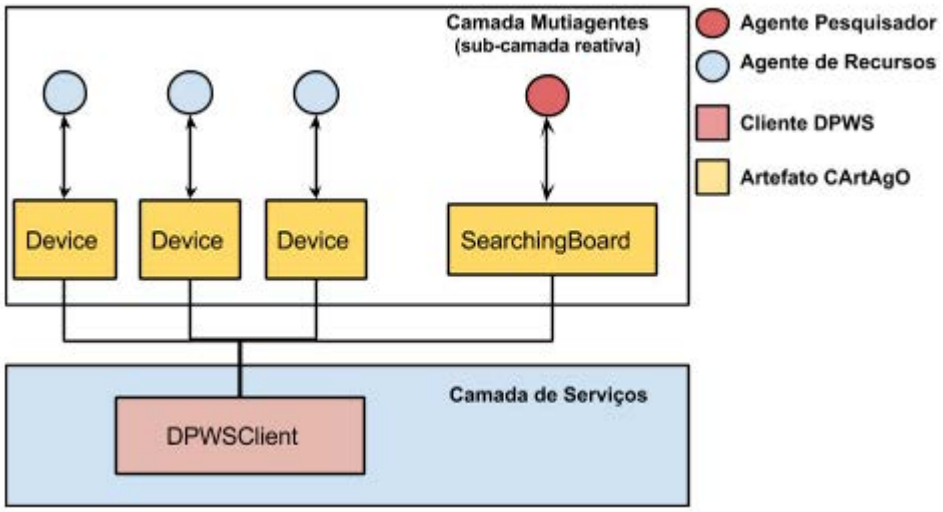

Figura 23: Artefatos CArtAgO que fazem interface com a camada de Serviços.

5.3.1 Implementação da Base de Conhecimentos

A Base de Conhecimentos, parte integrante da camada multiagentes, fornece as funcionalidades associadas ao gerenciamento do conhecimento da aplicação e é acessada pelo Agente Gerenciador de Conhecimentos.

A base de conhecimentos é responsável por armazenar o conhecimento adquirido pelos agentes durante a execução do sistema. Este conhecimento é estruturado na forma de ontologias que representam um conjunto de conceitos. Os conceitos descritos na base de conhecimentos estão relacionados às tecnologias necessárias usadas durante a execução dos requisitos funcionais da arquitetura, ou seja, conceitos relacionados às tarefas, às regras da organização, aos dispositivos DPWS e aos contextos.

As ontologias de aplicação foram desenvolvidas na linguagem OWL, com o auxílio da ferramenta Protégé. No sistema, a interface com a Base de Conhecimentos com os agentes Jason é implementada a partir de um artefato CArtAgO, que encapsula funcionalidades do *framework* Apache Jena relacionadas ao gerenciamento e a consulta das ontologias. Este artefato permite ao Agente Gerenciador de Conhecimento realizar as operações de consulta por meio da linguagem SPARQL. Além disso, o artefato representa um *middleware* que permite a tradução do conhecimento armazenado em arquivos RDF ou OWL para a linguagem *AgentSpeak*.

Os conceitos associados com dispositivos foram descritos na ontologia e representam o mapeamento dos metadados utilizados pela tecnologia DPWS e pelo *framework* JMEDS, representando na tabela 2 os dispositivos, seus serviços e operações.

Tabela 2: Descrição das classes da ontologia que descrevem os conceitos associados à tecnologia DPWS.

Classe	**Descrição**
Device (Dispositivo)	Representa um dispositivo DPWS conectado à rede local.
Service (Serviço)	Representa os serviços que o dispositivo DPWS apresenta.
Operation (Operação)	Representa as operações que podem ser executadas no dispositivo a partir de um dado serviço.
OperationInputParameter (Parâmetro de entrada da operação)	Parâmetros de entrada que uma dada operação pode apresentar. Um parâmetro de entrada apresenta um nome e tipo de dado.
Variable (Variável)	Representa variáveis de monitoramento e controle

Na tabela 3 são apresentadas as relações entre os conceitos associados à tecnologia DPWS.

Tabela 3: Descrição das relações entre as classes da ontologia que descrevem os conceitos associados a tecnologia DPWS.

Domínio	**Relação**	***Range***	**Descrição**
Device	hasService	Service	Usada para representar os serviços associados a um dado dispositivo
Service	hasOperation	Operation	Associa as operações que estão presentes em um dado serviço
Operation	hasInputParameter	OperationInputParameter	Associa uma operação do dispositivo a um parâmetro de entrada.
Operation	canMonitorVariable	Variable	Associa uma variável medida pelo dispositivo a uma operação. Neste caso, o dispositivo que apresenta tal operação é caracterizado como sensor.

As classes "Device" e "OperationInputParameter" também apresentam propriedade de dados (*data properties*). Tais propriedades são disponibilizadas nos metadados dos dispositivos segundo a especificação do DPWS. A classe "Device" apresenta três propriedades denominadas "friendlyName", "model" e "manufacturer", que representam um nome do dispositivo, o seu modelo e o seu fabricante. Já a classe "OperationInputParameter" apresenta propriedades de dados relacionadas ao tipo do parâmetro. Isso é necessário para permitir aos Agentes de Recurso a correta passagem de parâmetros durante a execução das operações.

Uma operação (Operation) é realizada sobre uma variável (Variable) e possui um parâmetro de entrada (OperationInputParameter). Por exemplo, um ar-condicionado pode fazer o controle e o monitoramento da variável temperatura, para isso, as operações que ajustam os valores desejados de temperatura devem receber um valor de entrada.

Além da descrição dos dispositivos segundo a especificação DPWS, outro conjunto de conceitos está relacionado à organização de agentes e à execução das tarefas do sistema (Figura 24).

Estes conceitos descrevem as relações que existem entre as dimensões estrutural (classe "StructuralSpecification"), funcional (classe "FunctionalSpecification") e normativa (classe "NormativeSpecification") da organização (classe "Organization") segundo as especificações utilizadas pelo *framework* Moise.

A dimensão normativa (Norm) relaciona a dimensão estrutural, representada pelos grupos (GroupSpecification) ou subgrupos (Subgroup) de papeis (Role), com a dimensão funcional. A dimensão funcional é representada pelos conjuntos de missiões (classe "Mission") e planos (classe "Scheme"). Serão estes os planos utilizados pelo Agente Gerenciador da Organização e pelos Agentes de Recurso que gerenciarão a execução das operações dos diversos dispositivos conectados.

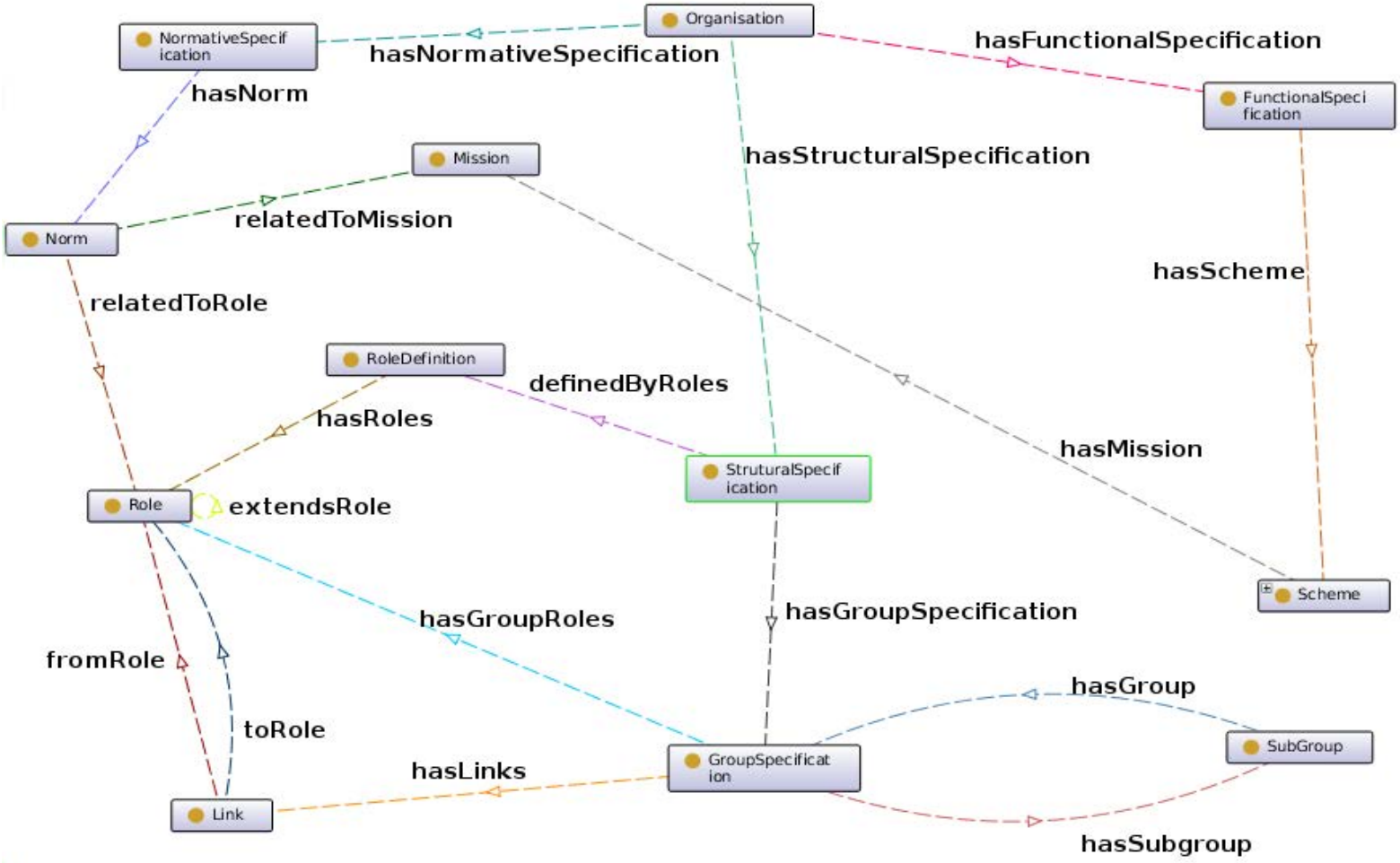

Figura 24: Conceitos da ontologia de aplicação relacionados à organização.

Este conjunto de conceitos é também utilizado na criação do arquivo XML de configuração do *Moise*. Isso ocorre quando todo o conhecimento descrito das especificações da organização (Figura 24) é transcrito de conceitos OWL para elementos XML. A construção das classes via JAXB[11] foi possível a partir do arquivo XML-Schema da especificação do Moise[12]. Dessa forma, novas especificações de organizações podem ser criadas automaticamente após consultas nas ontologias.

Por fim, o último conjunto de conceitos presentes na base de conhecimentos está relacionado ao contexto de tempo e de identificação, dados na tabela 4.

Tabela 4: Descrição das classes da ontologia que descrevem os conceitos associados aos contextos tratados no estudo de caso.

Classe	Descrição
Context (Contexto)	Esta classe representa um tipo de contexto. No estudo de caso foram considerados contextos de tempo e de identidade. Todo contexto apresenta um nome que o identifica. Todas as demais classes relacionadas nesta tabela são subclasses desta classe.
IdContext (Contexto de Identidade)	Esta classe representa um contexto de identidade cujo objetivo é identificar um usuário ou informar que uma tarefa deve ser realizada automaticamente. O usuário é reconhecido a partir de um nome.
TimeContext (Contexto de Tempo)	Esta classe representa um contexto de tempo. Todas as demais classes relacionadas nesta tabela são subclasses desta classe e representam tipos específicos de contextos de tempo.
EveryTime (A todo instante)	Esta classe representa um contexto de tempo em que a cada intervalo de tempo, expresso em horas e minutos, o sistema executará a tarefa.
TimeDependent (Dependente de um horário)	Esta classe representa um contexto de tempo em que uma tarefa será executada em um determinado horário diariamente.
WeekDependent (Dependente de um dia da semana)	Representa um contexto em que a tarefa será executada em um determinado dia da semana. Neste caso, além do horário, esta classe também apresenta um dia da semana em que a tarefa será executada.

[11] https://jaxb.java.net/

[12] http://moise.sourceforge.net/xml/os.xsd

DateDependent (Dependente de uma data)	Representa um contexto em que a tarefa será executada em uma determinada data. Neste caso, além do horário, esta classe também apresenta dia, mês e ano.

O contexto ("Context") descreve um tipo de contexto que, no caso da aplicação, podem ser contexto de tempo (classe "TimeContext") ou de identificação (classe "IdContext"). O contexto também está relacionado ao "Scheme", definido como o conjunto de planos que os agentes da organização executarão durante uma tarefa executada por toda organização. Isso significa que uma dada tarefa definida por um conjunto de planos ("Scheme") pode ser executada por algo (no caso o próprio sistema), ou alguém (no caso um dos usuários do sistema), e também em um instante ou intervalo de tempo definido.

. Existem diferentes tipos de contextos de tempo (TimeContext). Neste caso, uma tarefa pode ser executada em uma data (classe "DateDependent"), em um horário (classe "TimeDependent"), em um dia da semana (classe "WeekDependent") ou mesmo a todo instante após um intervalo de tempo (classe "EveryTime") em minutos ou horas.

Estas classes também apresentam propriedades de dados associadas à data, horas e intervalos de tempo.

As classes principais que relacionam os três conjuntos de conceitos apresentados (dispositivos, organização e contextos) estão representadas a seguir:

O “Scheme" é o principal conceito que representa a execução de uma tarefa por parte do sistema e se relaciona aos contextos de tempo ("TimeContext"), identidade ("IdContext"), "Mission" e "Operation". Tais relações são necessárias para definir quando, quais agentes, como e com quais dispositivos uma dada tarefa será executada pela organização de agentes do sistema.

A partir das informações de indivíduos da classe "TimeContext" é possível obter informações a respeito de quando e com qual periodicidade a tarefa será executada. Já indivíduos da classe "IdContext" podem informar ao sistema se a tarefa será executada automaticamente ou a partir de uma requisição de um usuário.

Uma vez que a tarefa foi requisitada ou disparada automaticamente pelo sistema, é necessário identificar os papéis dos agentes (classe "Role") e as missões (classe "Mission") que cada agente irá executar. Neste contexto, uma missão é um conjunto de objetivos que devem ser atingidos pelos agentes da organização. Além disso, os objetivos de cada agente estão associados às operações dos dispositivos conectados à rede local (Figura 25).

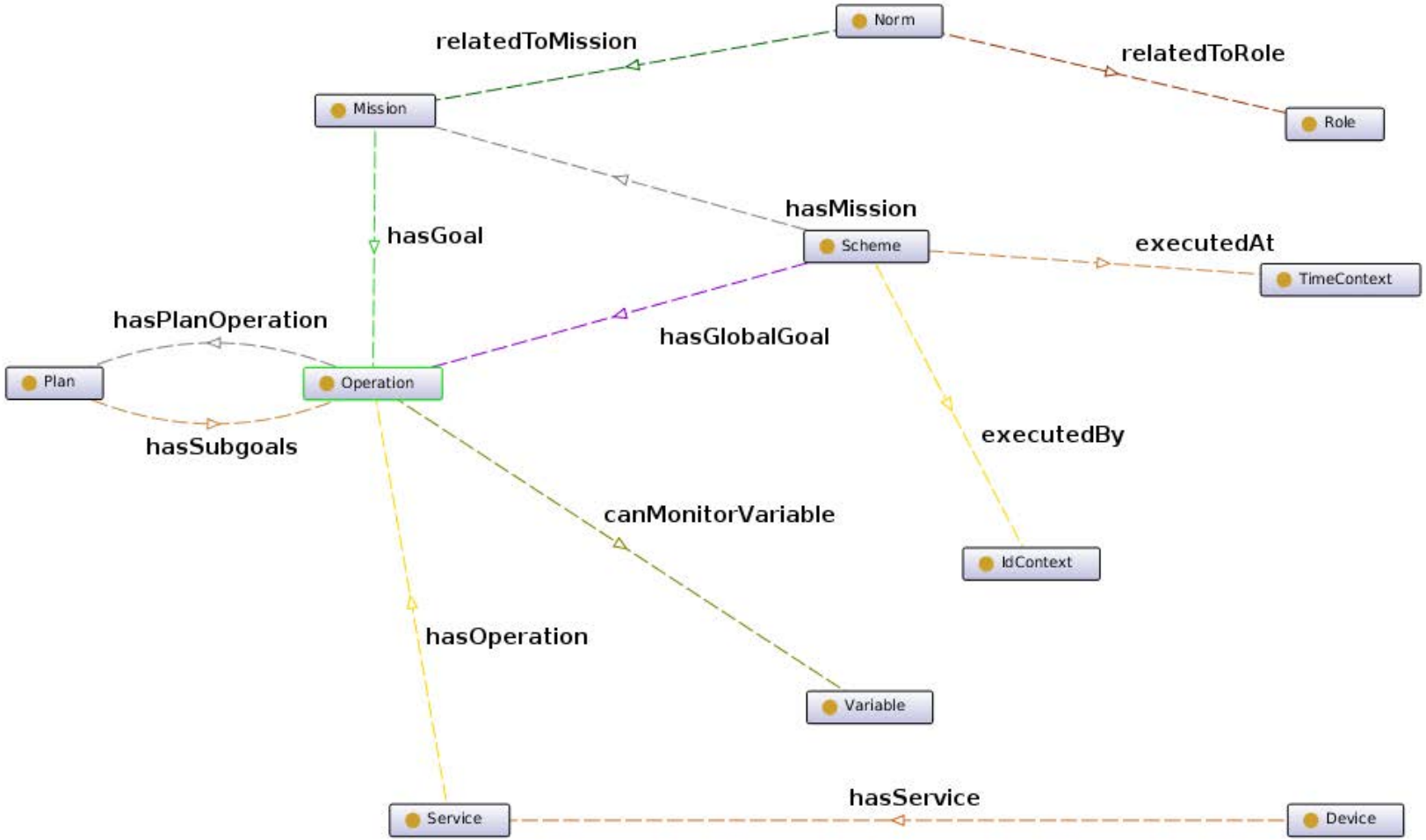

Figura 25: Conceitos da ontologia de aplicação que relacionam os diferentes domínios.

A realização da tarefa está relacionada à ordem e à sequência com que os objetivos devem ser atingidos. A tarefa é composta por missões (“Mission”), que por sua vez são compostas por um conjunto de planos ("Plan"). Os planos serão executados pelos agentes a partir das operações (“Operation") presentes nos dispositivos. Tais conceitos representam o mapeamento da árvore de decomposição de objetivos usada pelo *framework* Moise (vide seção 2.2.4.3).

5.3.2. A implementação da organização

A implementação da organização de agentes foi feita utilizando o *framework* Moise. A especificação da organização é feita através da descrição dos aspectos estruturais, funcionais e normativos das organizações.

Para auxiliar na especificação dos papéis dos agentes e suas interações é utilizado o diagrama OML oferecido pelo *framework* JaCaMo. Na Figura 26 é apresentada, em OML, a organização estrutural dos agentes em função dos papéis desempenhados.

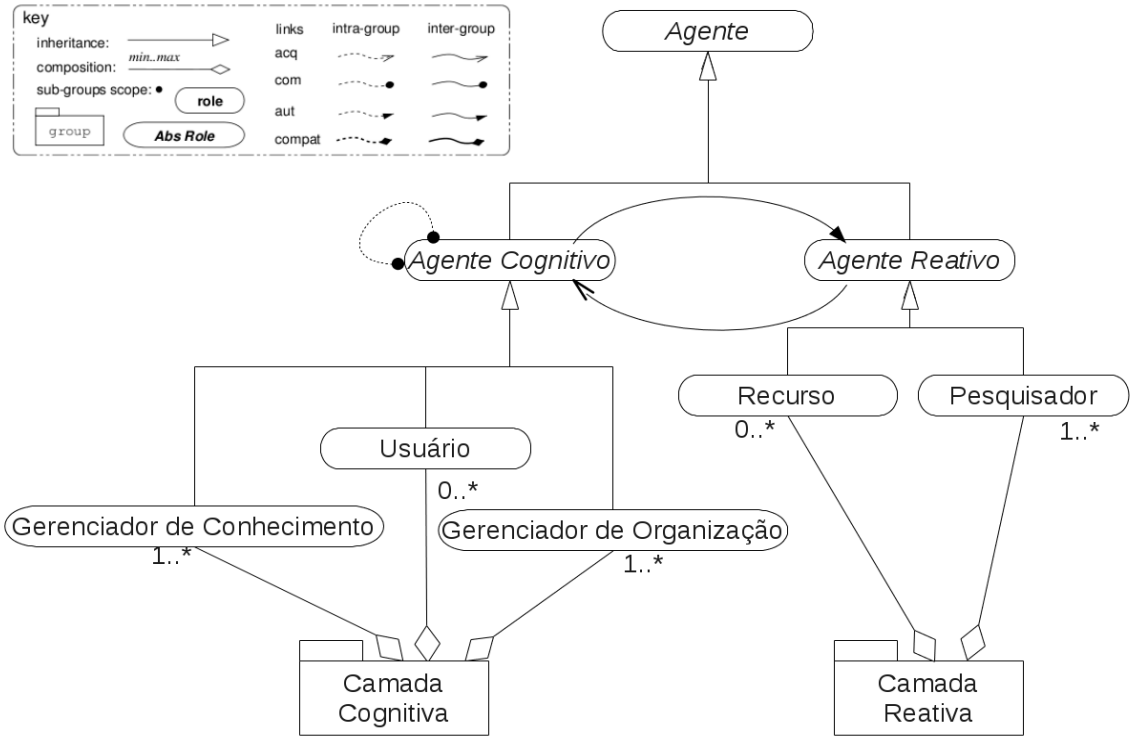

Figura 26: Nível estrutural da organização dos agentes descrita em diagrama OML.

No nível estrutural da organização existem dois grupos que representam as camadas cognitiva e reativa. Para auxiliar o entendimento do diagrama, dois papeis abstratos denominados "Agente Cognitivo" e "Agente Reativo" são definidos. Estes papeis permitem especificar as relações de autoridade (*authority*) e informação (*acquaintance*) existente entre os agentes da Camada Cognitiva e os agentes da Camada Reativa.

Segundo a descrição conceitual apresentada na seção 4.2, durante a execução da aplicação podem existir vários agentes desempenhando o papel de Agente de Recursos, dependendo do número de dispositivos conectados; ou mesmo nenhum agente, caso nenhum dispositivo foi detectado. Da mesma forma, o número de agentes desempenhando o papel de Agente de Usuário depende do número de usuários utilizando o sistema.

A organização de agentes é constituída pelos agentes do sistema, inicialmente formada pelos papéis de Agente Gerenciador da Organização, Agente Gerenciador de Conhecimento e o Agente Pesquisador. Os demais agentes são adicionados dinamicamente assim que os novos recursos sejam identificados e integrados ao sistema adotando o papel de Agentes de Recursos. O gerenciamento das três dimensões da organização é realizado a partir de artefatos da infraestrutura ORA4MAS.

Na infraestrutura ORA4MAS, dois artefatos, denominados *GroupBoard* e *SchemeBoard* são encarregados de gerenciar as dimensões estruturais, funcionais e normativas da organização (Figura 27).

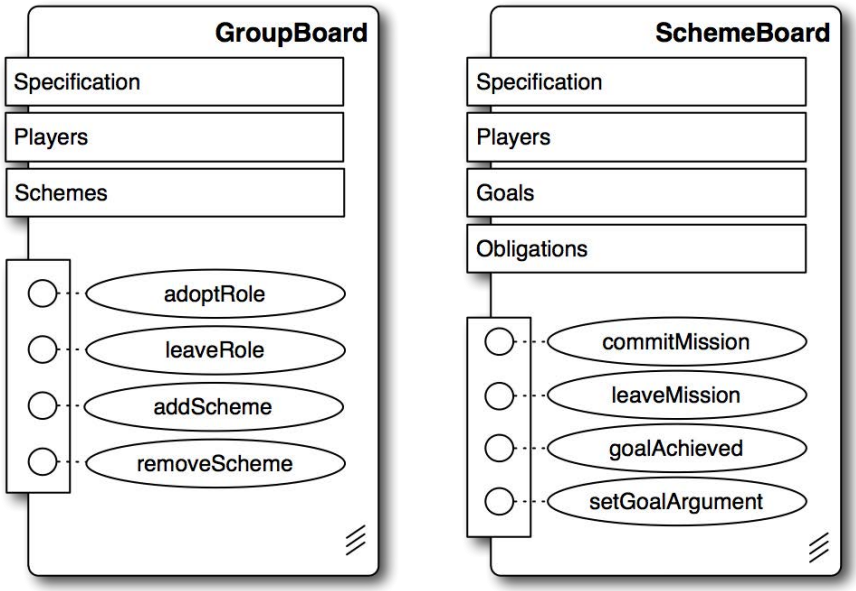

Figura 27: Artefatos ORA4MAS.
Fonte: Hübner, Sichman e Boissier, (2002).

Estes artefatos são utilizados pelo Agente Gerenciador da Organização e por todos os demais agentes do sistema com finalidade de fornecer funcionalidades relacionadas ao gerenciamento e garantir que as normas da organização sejam executadas pelos demais agentes. Em particular o artefato *GroupBoard*, apresenta funcionalidades relacionadas ao gerenciamento da organização, é o artefato utilizado somente pelos agentes que assumem o papel de Agente Gerenciador da Organização, para controlar a execução de tarefas que envolvam os demais agentes.

O artefato *SchemeBoard* é utilizado não somente pelo Agente Gerenciador da Organização como também pelos Agentes de Recurso. Este artefato apresenta as funcionalidades de controle da execução das tarefas executadas pela organização. O controle da execução das tarefas está definido no nível funcional da organização. De maneira geral este artefato controla a sequência de objetivos, previamente especificada na dimensão funcional da organização, os quais deverão ser executados e atingidos por cada Agente de Recurso que participa da tarefa a ser executada. Por fim, a dimensão normativa definida para cada um dos agentes da organização é apresentada aos agentes desempenhando o papel de Agente Gerenciador da Organização e Agente de Recursos do sistema a partir da propriedade observável "*obligations*", presente no artefato *SchemeBoard*.

A descrição de cada plano para cada um dos papéis implementados, pode ser consultada no apêndice A.

6 ESTUDO DE CASO

Um estudo de caso utilizando a arquitetura foi implementado para a execução dos cenários descritos na seção 4.2.2 (inicialização, descoberta de dispositivos, determinação do contexto, gerenciamento de dispositivos e interação entre o usuário e o sistema).

Deste modo, para avaliar a viabilidade desta proposta, um protótipo de um sistema foi elaborado de modo a simular o gerenciamento de dispositivos no ambiente da IoT. O ambiente da Internet das coisas simulado consiste nos dispositivos presentes em uma casa inteligente. Dessa forma, os resultados apresentados nesta seção representam um conjunto de cenários de simulação de domótica - automação residencial inteligente. No estudo de caso é dado ênfase ao uso de sistemas multiagentes aliado às ontologias na implementação de uma aplicação no referido contexto.

6.1 Simulação do ambiente da Internet das Coisas

Em razão da falta de recursos para implementar uma estrutura física, foi implementado um simulador de um ambiente da Internet das coisas. O domínio escolhido é o da automação residencial. Como o objetivo do estudo de caso não é avaliar questões de desempenho, justifica-se a adoção de um cenário onde o tempo de reação e execução de dispositivos não seja tão crítico. No contexto da automação residencial a escala de tempo em que ocorrem a execução de tarefas a partir do sistema se dá em tempos superiores à minutos.

O ambiente foi simulado a partir de dispositivos baseados na tecnologia DPWS. A simulação dos dispositivos reais é implementada em Java a partir da API fornecida pelo *framework* JMEDS (Figura 22). Os dispositivos usados no ambiente simulado são dados a seguir:

- Ar-condicionado;
- Cafeteria;
- Forno elétrico;
- Notebook;
- Sensor de temperatura;

Os serviços e operações implementadas para cada um dos dispositivos utilizados no estudo de caso são descritos no apêndice B.

Os cenários de execução para este estudo de caso, descritos conceitualmente na seção 4.2.2, dependem da Base de Conhecimentos onde estão armazenadas as ontologias que serão detalhadas a seguir.

6.2 Ontologias

O conhecimento necessário para descrever os dispositivos e as tarefas foi descrito em ontologias. Nas ontologias da Base de Conhecimentos do sistema foram descritas três tarefas que representam o sistema em três possíveis estados de execução: tarefas automatizadas executadas de forma contínua a cada intervalo de tempo, tarefas automatizadas executadas de forma agendada em um instante de tempo, e tarefas solicitadas pelo usuário. Exemplos dessas tarefas são "controlTemperatureSch", "dailyMorningSch" e "turnAllOffSch".

A tarefa denominada "controlTemperatureSch", descrita na figura 28, representa o controle da temperatura ambiente da casa, sendo necessários para este fim dispositivos que possam obter o valor da temperatura (sensores) e dispositivos que possam controlar a temperatura ambiente até que esta atinja um dado valor desejado. Além dos recursos necessários, esta tarefa será executada (executedAt) de forma contínua a cada 15 minutos pelo sistema, pois esta tarefa está relacionada ao contexto de tempo "everytime" e de identidade (executedBy) "system" (Figura 28).

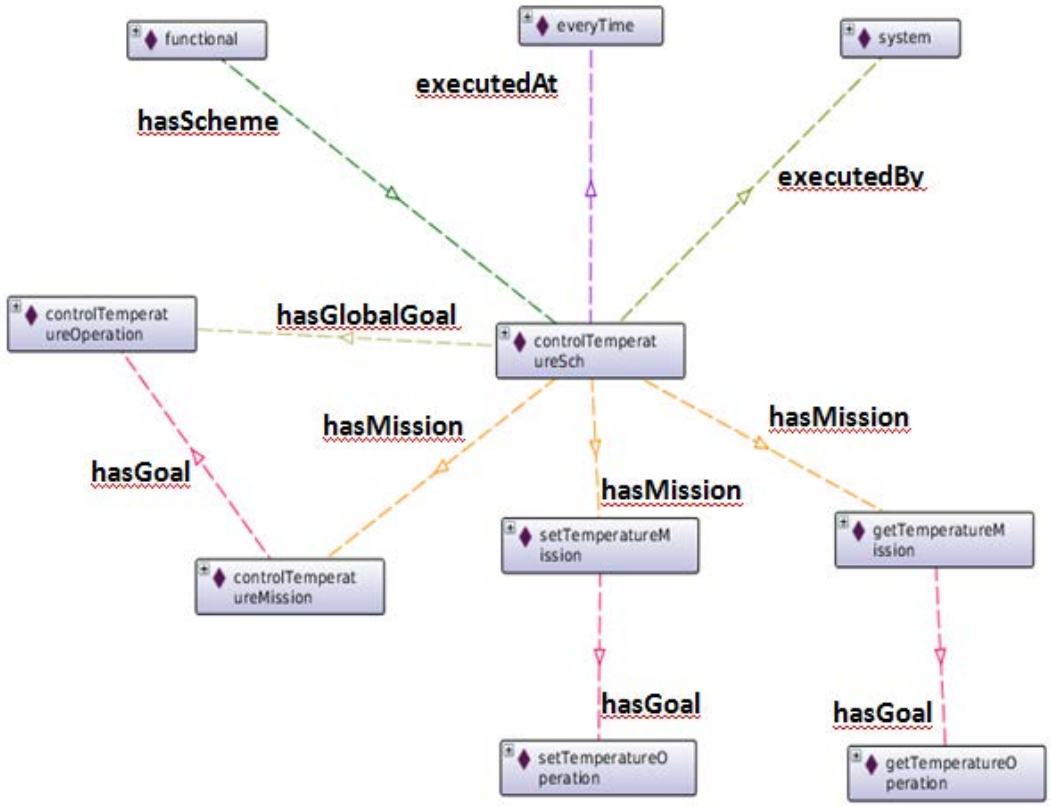

Figura 28: Descrição da tarefa "controlTemperatureSch".

Os demais conceitos relacionados na ontologia são as missões (conjuntos de objetivos a serem atingidos pelos agentes) e as operações que cada um dos dispositivos deve apresentar para possibilitar a execução desta tarefa. Em particular existem três missões (“controlTemperatureMission”, “setTemperatureMission” e “getTemperatureMission”). Estas missões relacionam o papel de Agente de Recursos à execução das operações dos dispositivos.

A outra tarefa denominada "dailyMorningSch envolve a execução automática segundo um instante de tempo previamente agendado (contexto "timedependentcontext"). Esta tarefa representa a execução de operações matinais comuns para um usuário da casa. Em particular esta tarefa será sempre executada em um determinado horário. Nesta tarefa o sistema sempre ligará e utilizará operações de três dos dispositivos presentes na casa, neste caso um notebook, um forno elétrico e uma cafeteira. A representação completa desta tarefa é dada na figura 29.

Esta tarefa, da mesma forma da anterior, também é executada automaticamente, pois está relacionada ao contexto de identidade "system". Além disso, o número de operações e missões relacionadas à esta tarefa é maior devido à sua maior complexidade envolvendo maior número de dispositivos e operações. Esta tarefa apresenta quatro missões “prepareOvenMission”, “prepareNoteMission”, “prepareCoffeMachineMission” e

“dailyMorningMission”. As missões apresentam um conjunto de operações relacionadas à ligar, executar uma operação e desligar cada um dos dispositivos.

Por fim, a tarefa "turnAllOfSch" é definida como a tarefa em que todos dispositivos, que apresentam operações relacionadas ao seu próprio desligamento e economia de energia, devem executar quando solicitada pelo usuário. Neste caso, esta tarefa está relacionada a um contexto de identidade "homeuser", pois não será executada automaticamente e sim quando o usuário desejar (Figura 30). .

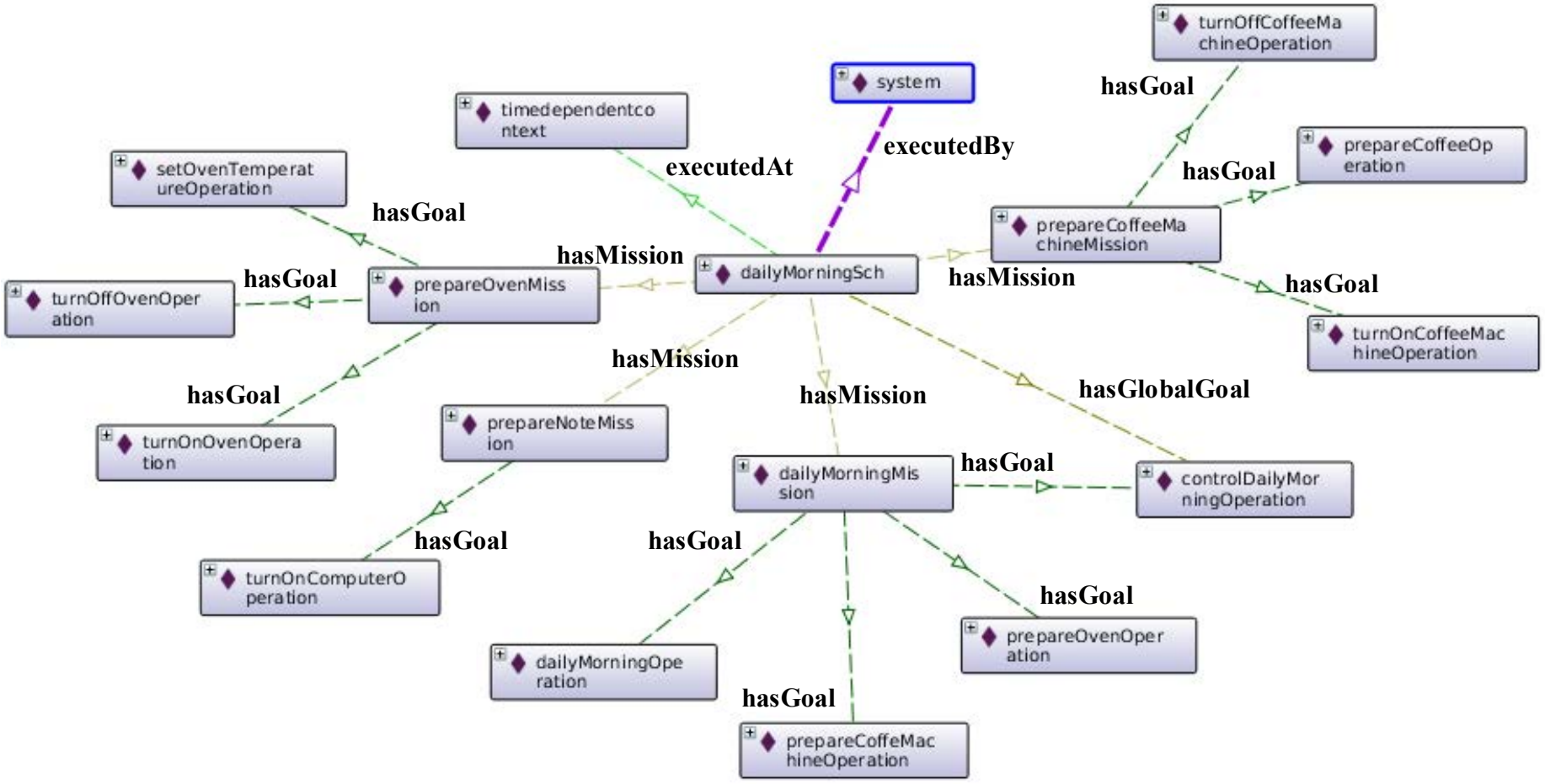

Figura 29: Descrição da tarefa "dailyMorningSch"

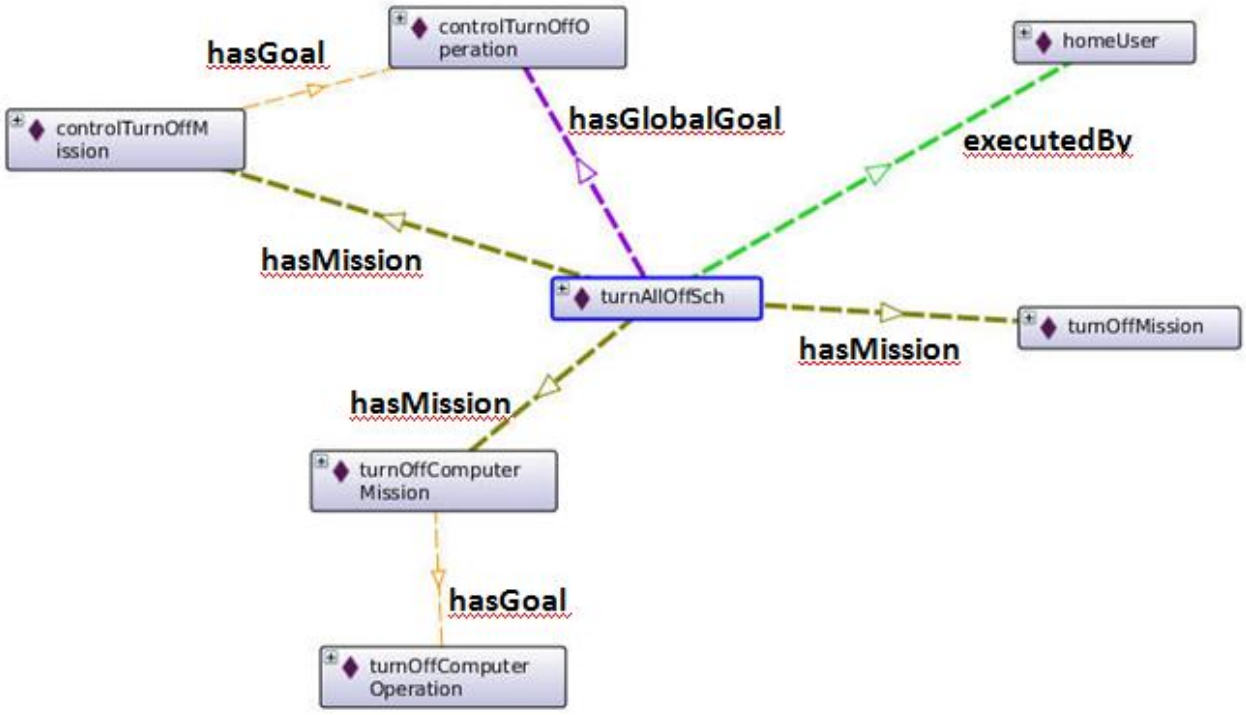

Figura 30: Descrição da tarefa "turnAllOffSch".

Os demais conceitos (missões e operações) estão relacionados às operações de desligamento dos dispositivos. Além disso, para esta tarefa não é definido um contexto de tempo, pois a execução desta tarefa será realizada no instante que ocorrer a requisição do usuário.

A seguir é apresentada a execução do protótipo em cada um dos cenários.

6.3 Cenários testados

Nesta seção é apresentada a execução dos principais componentes da arquitetura segundo os cenários definidos nas subseções da seção 4.2.2.

6.3.1 Inicialização dos agentes

Neste cenário são implementadas as ações realizadas pelos agentes no momento da inicialização do sistema.

Inicialmente, o Agente Gerenciador de Conhecimentos executa os planos para criar o artefato "KnowledgeBaseArtifact", carregar as ontologias em memória e permitir a criação do arquivo de configuração do *framework* Moise (apêndice C).

Paralelamente, o Agente Pesquisador inicia as interfaces da camada de serviços a partir da criação do artefato "SearchingBoard", que inicializa o simulador do ambiente implementado no *framework* JMEDS. Dessa forma, o sistema está pronto para receber mensagens, segundo a especificação WS-*Discovery*[13], dos novos dispositivos conectados. A partir de tais mensagens, o sistema poderá receber os metadados dos dispositivos, que permitirão a sua identificação.

O Agente Gerenciador da Organização, inicializa a infraestrutura da organização de agentes, criando um artefato "GroupBoard" e solicita aos outros agentes para desempenhar as suas funções na organização (Figura 31).

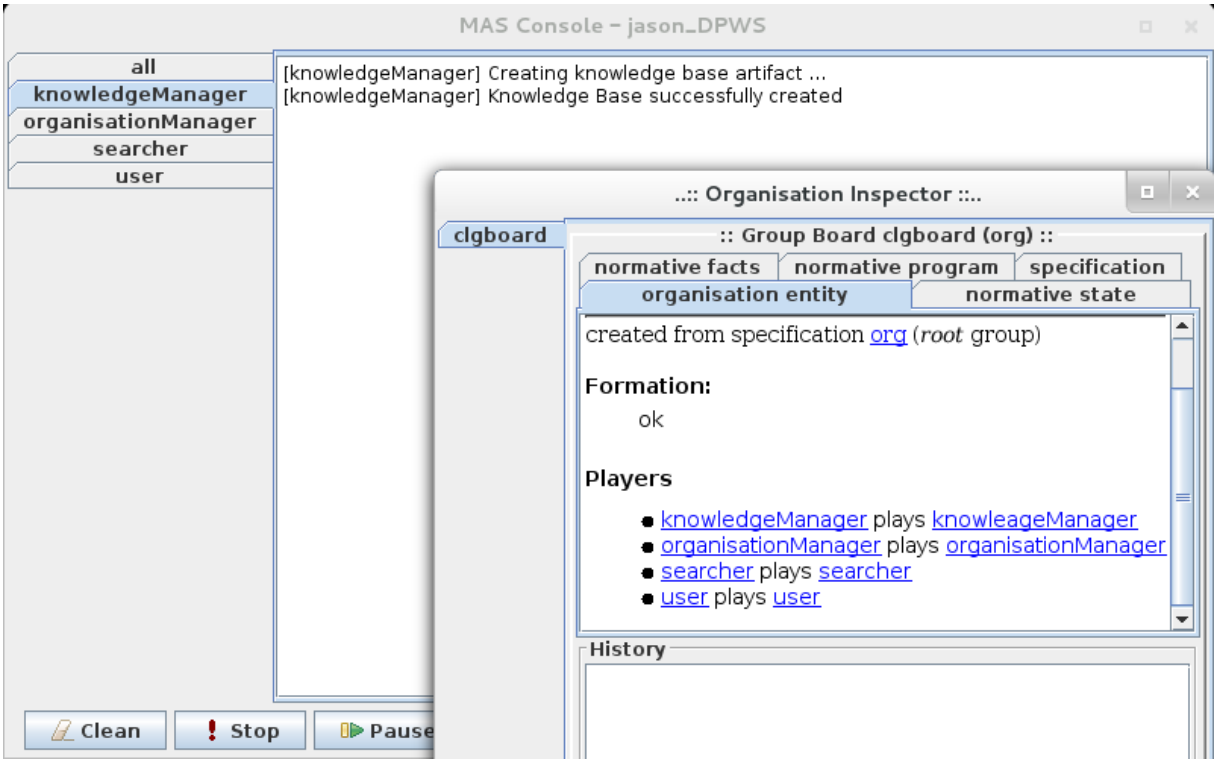

Figura 31: Mensagens da interface do sistema durante a inicialização dos agentes.

Na Figura 30 é apresentada a interface do sistema com as mensagens de saída do sistema multiagentes e o inspetor da organização, interface gráfica disponível a partir do

13 http://docs.oasis-open.org/ws-dd/discovery/1.1/os/wsdd-discovery-1.1-spec-os.html

framework Moise. O inspetor da organização demonstra que os agentes inicializados no sistema assumiram um papel na organização. Além disso, são apresentadas as mensagens do Agente Gerenciador de Conhecimentos, informando que a Base de Conhecimentos está pronta para receber requisições. Finalmente, o "Agente de Usuário" inicializa a interface Web para receber requisições HTTP (Figura 32). .

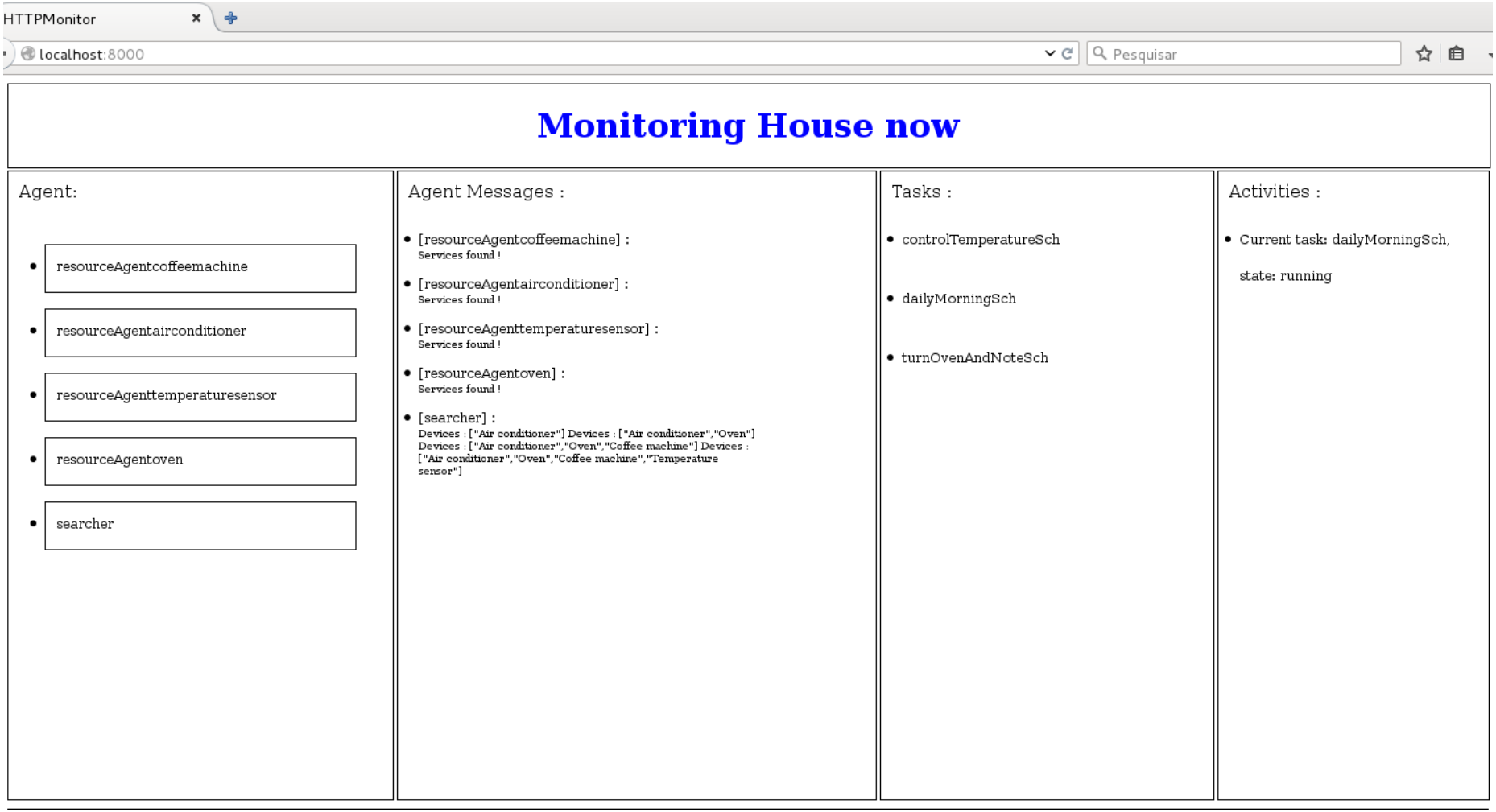

Figura 32: terface WEB disponível para os usuários do sistema

Na interface Web, o usuário obtém informações sobre os agentes do sistema (Agents), suas mensagens (Agent Messages), as tarefas disponíveis para a execução a partir do sistema (Tasks) e a tarefa que está gerenciada executada no momento (Activities). Além disso é possível verificar o estado da tarefa (executando, parada ou aguardando).

6.3.2 Identificação dos recursos

Neste cenário é implementada a execução da identificação dos recursos (dispositivos serviços e operações). O objetivo deste cenário é a identificação dos metadados, serviços e operações do novo dispositivo conectado.

A descoberta de dispositivos é realizada sempre que um novo dispositivo é conectado à rede. Quando um novo dispositivo entra na rede, o Agente Pesquisador percebe sua entrada (Figura 33). Neste momento o agente recupera os metadados descritivos do dispositivo, e os envia para o Agente Gerenciador de Conhecimentos.

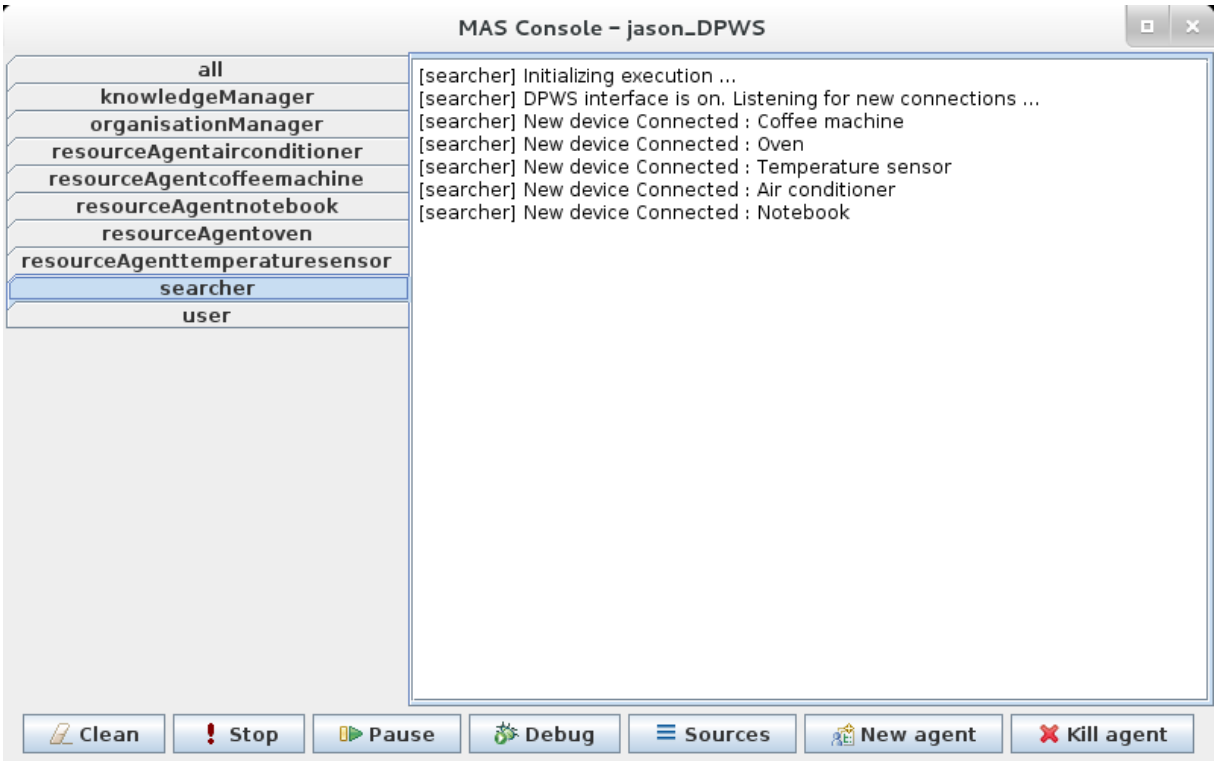

Figura 33: Mensagens do "Agente Pesquisador" ao identificar novos dispositivos conectados à rede.

O Agente Gerenciador de Conhecimentos irá executar a busca pelas informações semânticas utilizando estes metadados.

Na simulação do ambiente os diversos dispositivos presentes na casa são conectados à rede sequencialmente. Dessa forma, o Agente Pesquisador deve detectá-los independente do instante em que cada dispositivo vier a ser conectado na rede.

Na Figura 33 são apresentadas as mensagens do Agente Pesquisador ao identificar os dispositivos. Após este processo, o Agente Pesquisador cria um novo Agente de Recursos que irá obter os metadados dos serviços e das operações do novo dispositivo identificado.

6.3.3 Busca pela descrição semântica

O terceiro cenário é a busca pela descrição semântica dos serviços e operações do dispositivo. Este cenário ocorre imediatamente após o cenário anterior denominado "identificação dos recursos". Assim que um novo dispositivo é conectado à rede, o Agente Pesquisador detecta e recebe os metadados que descrevem informações gerais do dispositivo.

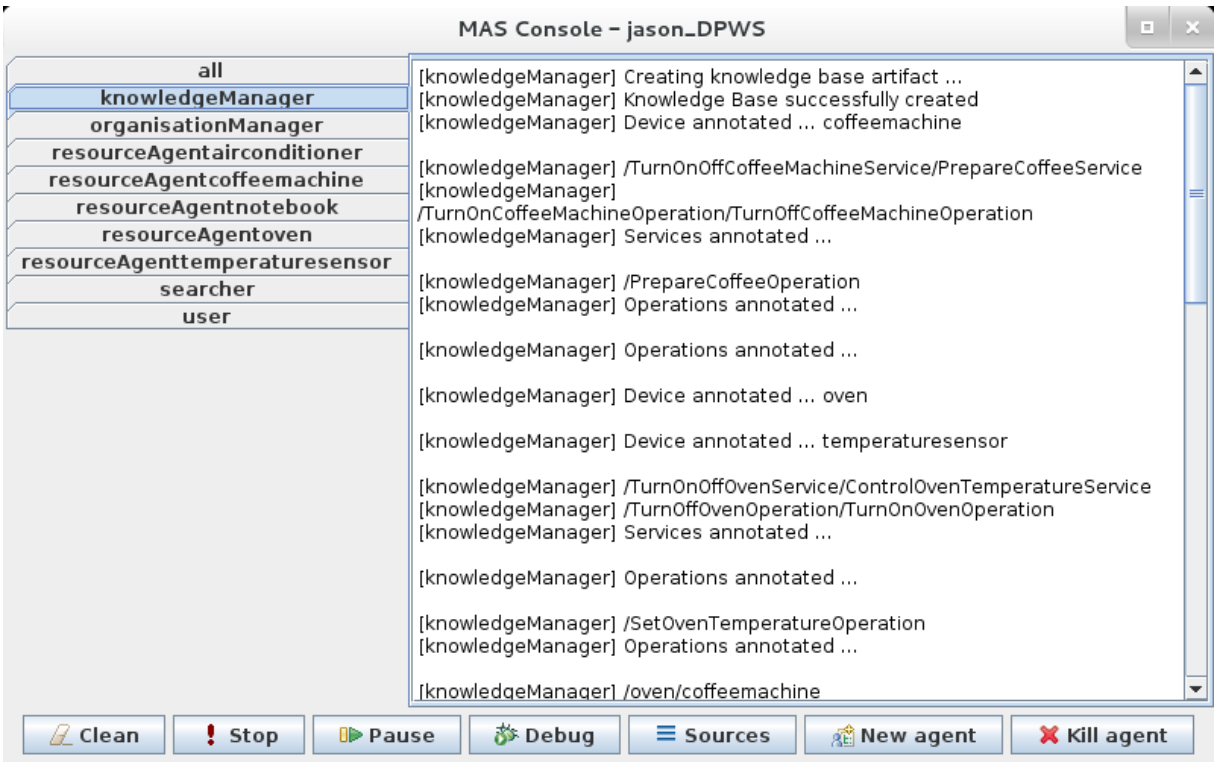

Figura 34: : Mensagens do "Agente Gerenciador de Conhecimentos" durante a execução do sistema. Em destaque a busca pela descrição semântica de dispositivos.

Os metadados do dispositivo são enviados para o agente Agente Gerenciador de Conhecimentos que realiza a consulta a Base de Conhecimentos para obter a descrição semântica do mesmo (Figura 34). Em seguida o Agente Pesquisador cria um novo Agente de Recursos de acordo com a descrição semântica.

O novo agente de Recurso criado recupera os metadados dos serviços e operações do dispositivo. Para cada nova operação recuperada do dispositivo, este agente criará um novo plano em sua base de planos que correspondem a novas ações que permitirão ao Agente de Recursos a execução das tarefas descritas na Base de Conhecimentos. Após a execução deste cenário, o sistema terá novos Agentes de Recursos e será capaz de executar as operações dos novos dispositivos (Figura 35).

As operações acima descritas serão executadas para cada um dos dispositivos que se conectarão ao sistema. É necessário comentar que novos agentes criados são responsáveis por gerenciar especificamente um dispositivo. Além disso, serão estes agentes que participarão da execução de tarefas gerenciadas pelo sistema.

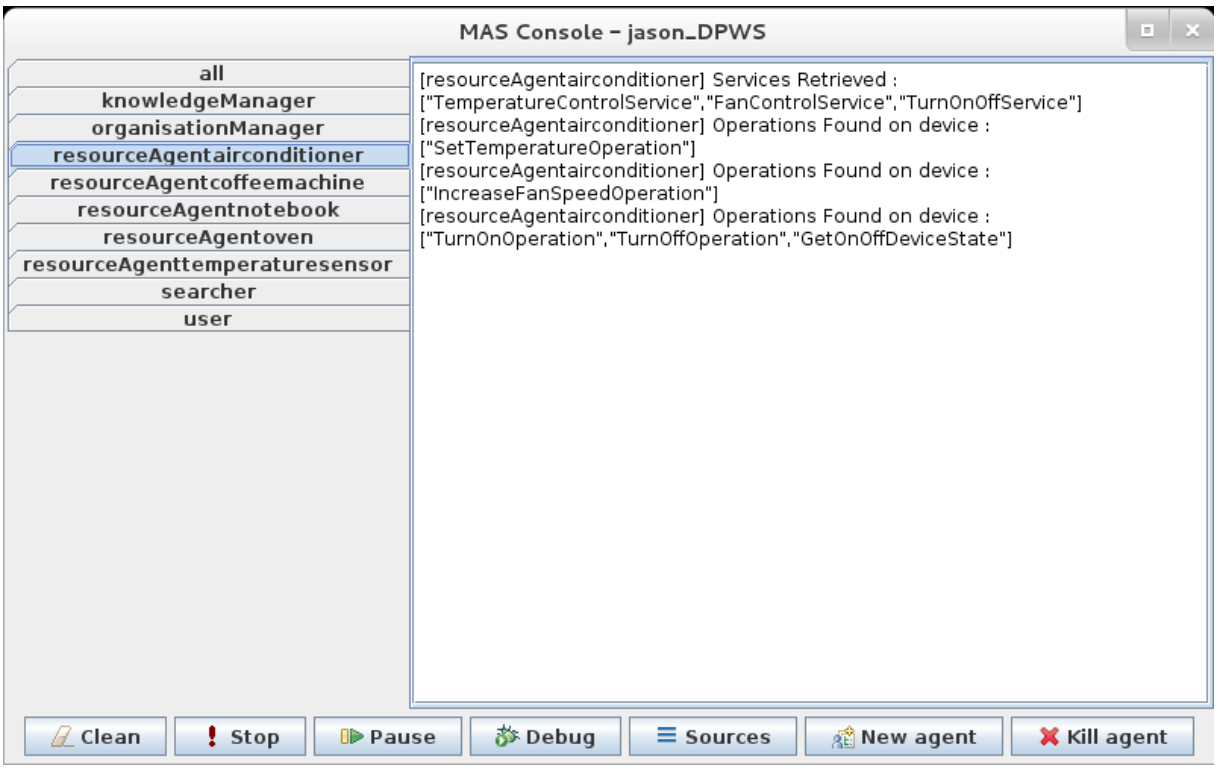

Figura 35: Interface com as mensagens do "Agente de Recursos" durante o cenário de busca pela descrição semântica de dispositivos.

6.3.4 Execução das tarefas

O quarto cenário está relacionado com a coordenação de diferentes Agentes de Recursos durante a execução de tarefas pelo sistema. Neste cenário participam os papéis de Agente de Recursos, o Agente Gerenciador da Organização e o Agente Gerenciador de Conhecimentos.

Este cenário inicia-se quando o Agente Gerenciador da Organização solicita ao Agente Gerenciador de Conhecimentos quais são as possíveis tarefas que podem ser executadas levando-se em consideração os recursos atualmente disponíveis (operações dos dispositivos conectados).

Após consultar a Base de Conhecimentos o Agente Gerenciador de Conhecimentos responde com a lista de possíveis tarefas que poderão ser executadas. Dessa forma, o Agente Gerenciador da Organização deve consultar agora em quais contextos (de tempo e identificação) esta tarefa deve ser realizada. Neste caso, existem duas possibilidades: o contexto é válido e a tarefa será executada ou o contexto não é válido e a tarefa deve ser executada em outro momento (Figura 36).

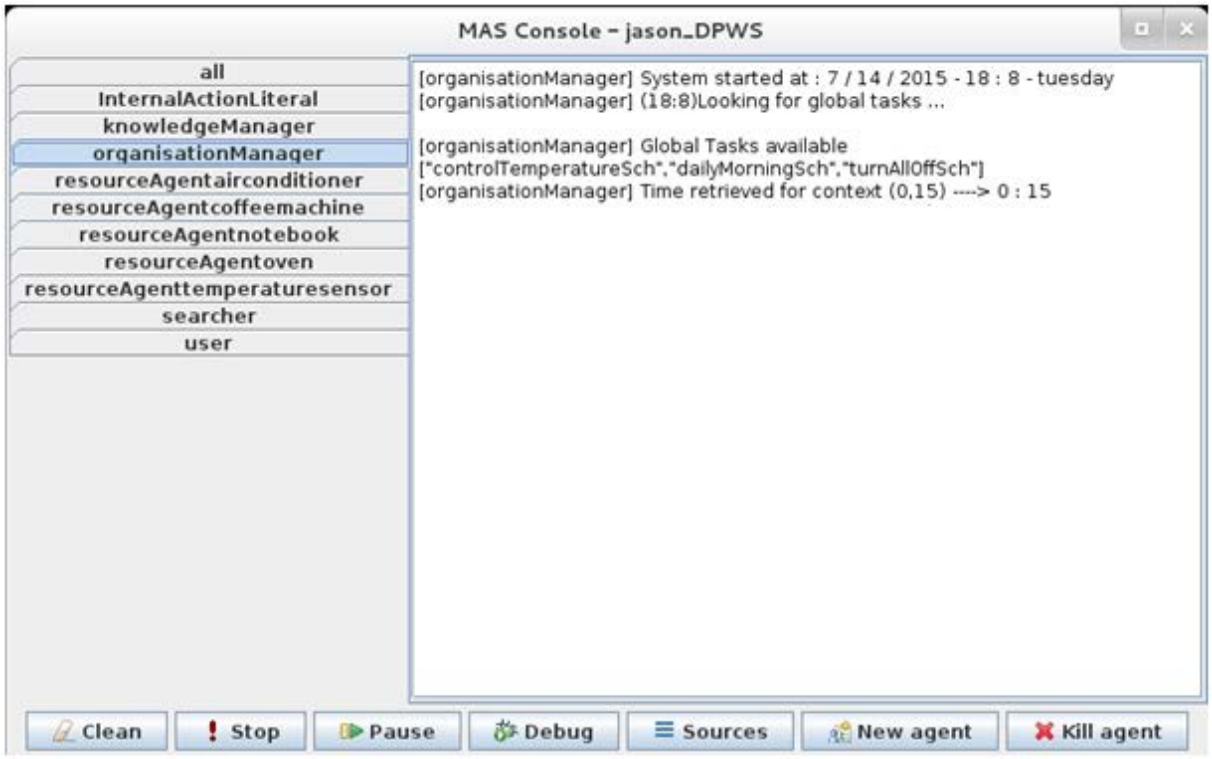

Figura 36: Mensagens do "Agente Gerenciador da Organização" durante a identificação de possíveis tarefas a serem executadas.

A Figura 36 demonstra que para cada nova tarefa disponível, o sistema busca sob quais contextos de tempo e identidade a tarefa deve ser executada. No exemplo, ilustrado na Figura 36, o sistema identificou as três tarefas descritas na Base de Conhecimentos. Dessa forma, o sistema realiza a verificação do contexto de tempo e determina que a tarefa "controlTemperatureSch" deve ser executada a cada 15 minutos.

Depois de obtidas tais informações, o Agente Gerenciador da Organização inicia a preparação da execução da tarefa enviando aos Agentes de Recursos os seus papéis na organização e suas missões (Figura 37).

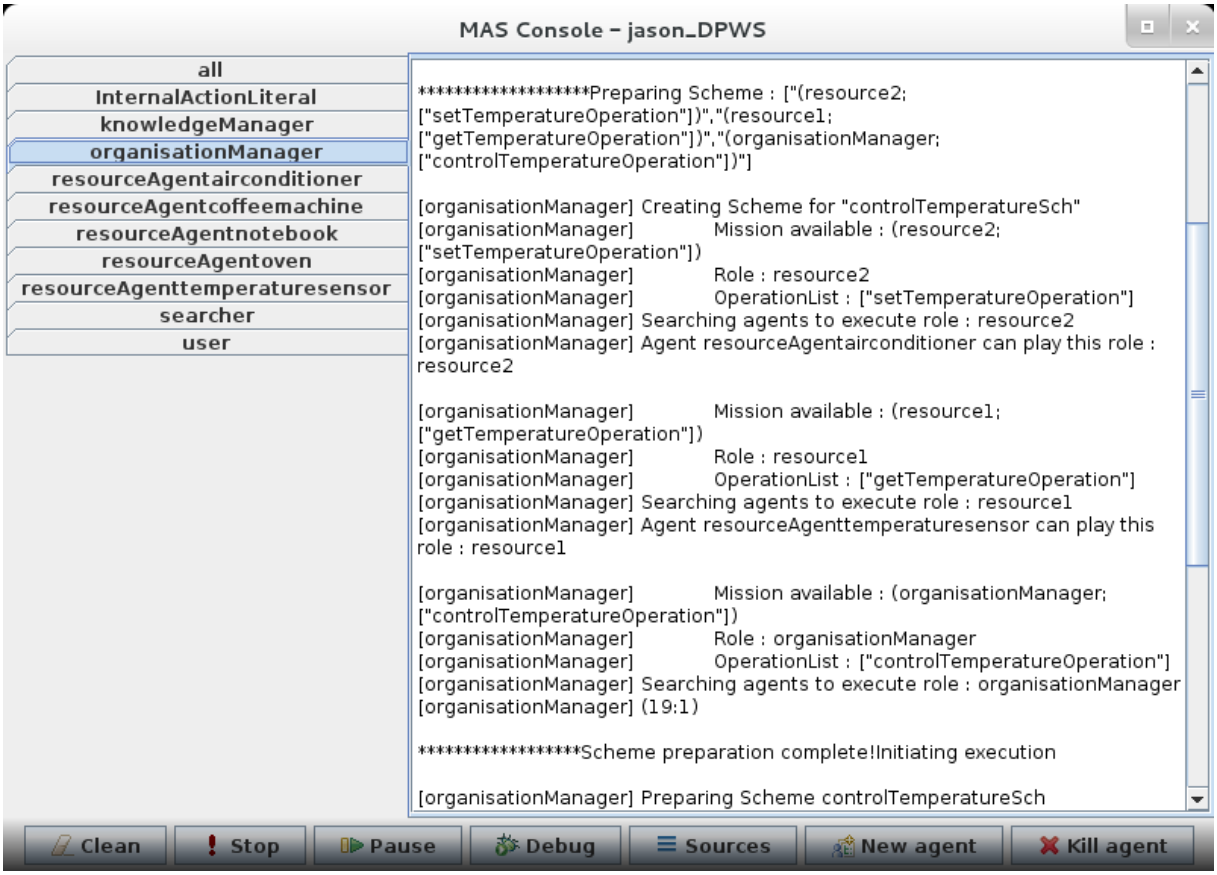

Figura 37: Mensagens do "Agente Gerenciador da Organização" durante a preparação de uma das tarefas.

No exemplo destacado na Figura 37, a tarefa denominada "controlTemperatureSch" está em fase de preparação, onde os "Agentes de Recursos" que gerenciam o dispositivo ar-condicionado e o sensor de temperatura devem assumir papéis na organização. Neste caso, de acordo com as operações executadas por cada dispositivo, é atribuído um papel temporário (resource1 e resource2) para cada um dos agentes. Estes papéis estão relacionados às missões “setTemperatureMission” e “getTemperatureMission” (Figura 28).

Uma vez identificados os Agentes de Recursos e atribuídos seus papeis, a tarefa é iniciada a partir da criação do artefato "SchemeBoard", e sua execução será gerenciada pelo próprio artefato. Cada objetivo demandado pela tarefa é disponibilizado pelo artefato na forma de sua propriedade observável "obligation". Estes objetivos devem ser alcançados agora pelos Agentes de Recursos de forma coordenada segundo a especificação funcional da organização (Figura 38).

Na Figura 38 o "Agente de Recursos" assume o papel "resource2" e se compromete a executar os objetivos associados à missão "setTemperatureMission". Após o termino da tarefa o agente deixa o papel temporário atribuído a ele na organização.

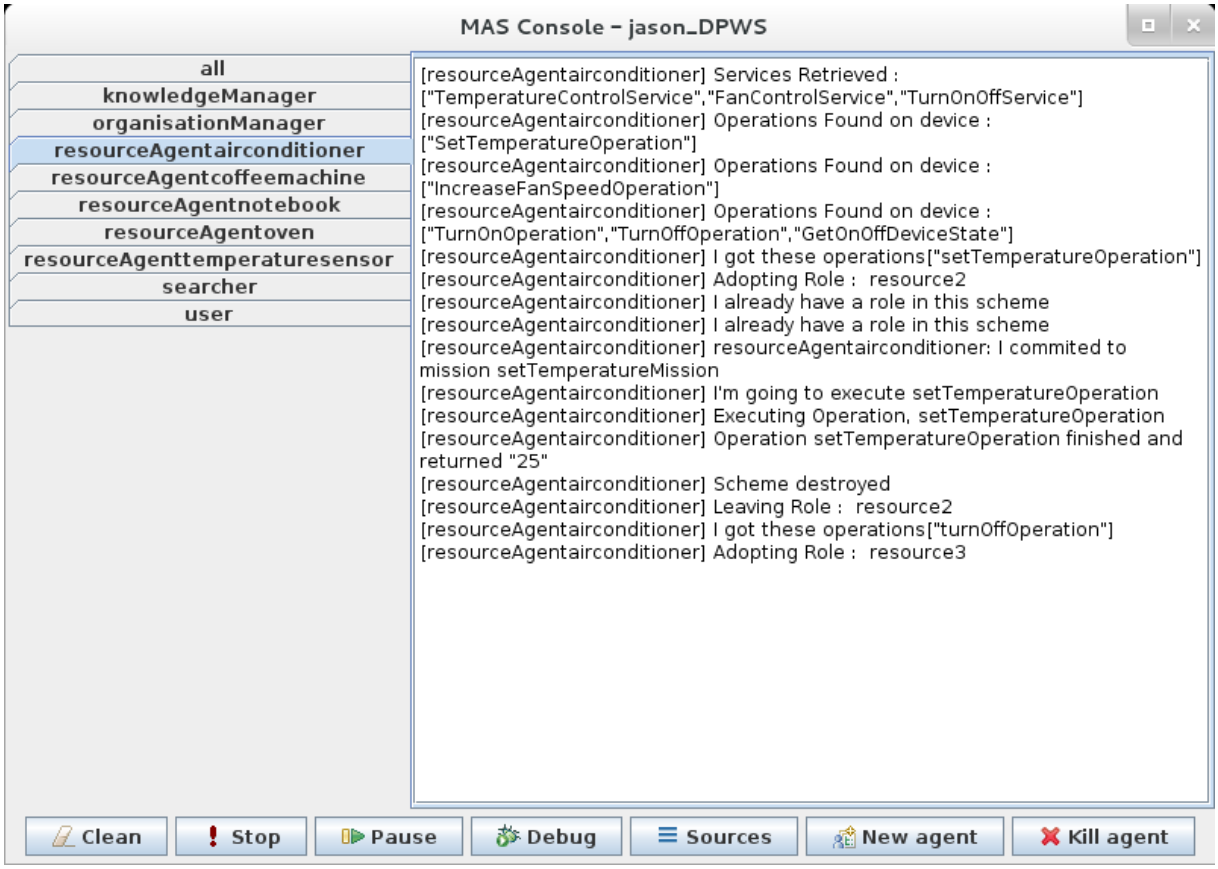

Figura 38: Mensagens do "Agente de Recursos" durante a execução de uma das tarefas.

Os objetivos que devem ser atingidos pelos Agentes de Recursos correspondem às operações presentes nos serviços do dispositivo, que em suma, correspondem às funções que o dispositivo executa sobre ambiente físico, por exemplo, ligar, desligar, controlar temperatura, abrir, entre outros. Cada serviço e operação dependerão do tipo de dispositivo usado pelo sistema durante a realização da tarefa.

Depois que os Agentes de Recursos adotaram seus respectivos papéis, a execução da tarefa é disparada e gerenciada pelo *framework* Moise e os artefatos ORA4MAS. Dessa forma, a função do Agente Gerenciador da Organização agora é verificar os eventos associados à tarefa e agir caso alguma anormalidade seja detectada. Se alguma anormalidade ocorrer durante a execução da tarefa, este agente deve destruir o artefato organizacional *SchemeBoard* e solicitar que todos os agentes envolvidos deixem de executar suas operações.

6.3.4.1. Execução automatizada contínua de tarefa

Para ilustrar a execução deste tipo de tarefa, será utilizada a tarefa "controlTemperatureSch", que tem por objetivo ajustar a temperatura da casa. Para a execução dessa tarefa o sistema deve apresentar como recursos um dispositivo que possa controlar a temperatura (no estudo de caso, um ar-condicionado) e um dispositivo sensor de temperatura.

A descrição dessa tarefa, segundo o diagrama OML, de como esta tarefa será realizada é dada na Figura 39.

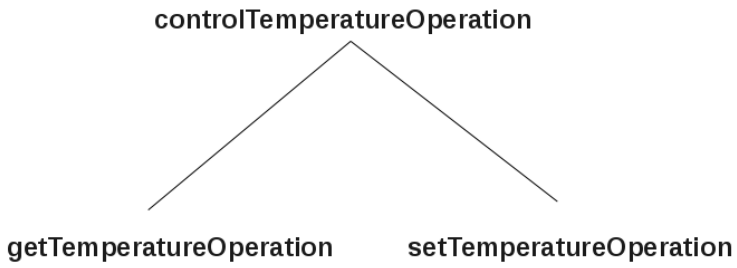

Figura 39: Descrição da tarefa "controlTemperatureSch" segundo o diagrama OML.

O diagrama acima ilustra que para atingir o objetivo global "controlTemperatureOperation", o sistema deve atingir os objetivos "getTemperatureOperation" seguido de "setTemperatureOperation". Neste caso esta tarefa somente será executada se existirem dispositivos que apresentem estes tipos de operações. Dessa forma, o Agente Gerenciador da Organização deve procurar por Agentes de Recursos, que gerenciem dispositivos que possibilitem a execução de tais operações. Depois de terminada a fase de preparação da tarefa, o Agente Gerenciador da Organização inicia a sua execução. A Figura 40 demonstra as mensagens do Agente Gerenciador da Organização durante a fase de execução desta tarefa.

Em destaque, na Figura 40, a execução da tarefa demonstrando a sequência de objetivos atingidos por cada um dos Agentes de Recursos e o objetivo atingido pelo Agente Gerenciador da Organização, que representa o término da execução da tarefa. Além disso, é possível observar que cada um dos Agentes de Recursos atingiram seus objetivos executando a operação correspondente de acordo com a especificação funcional da organização. Em particular, destaca-se que a execução da operação “getTemperatureOperation” retornou como saída o valor de 26,84, e a execução da operação “setTemperatureOperation” retornou o valor 25, simulando o ajuste de temperatura ambiente da casa.

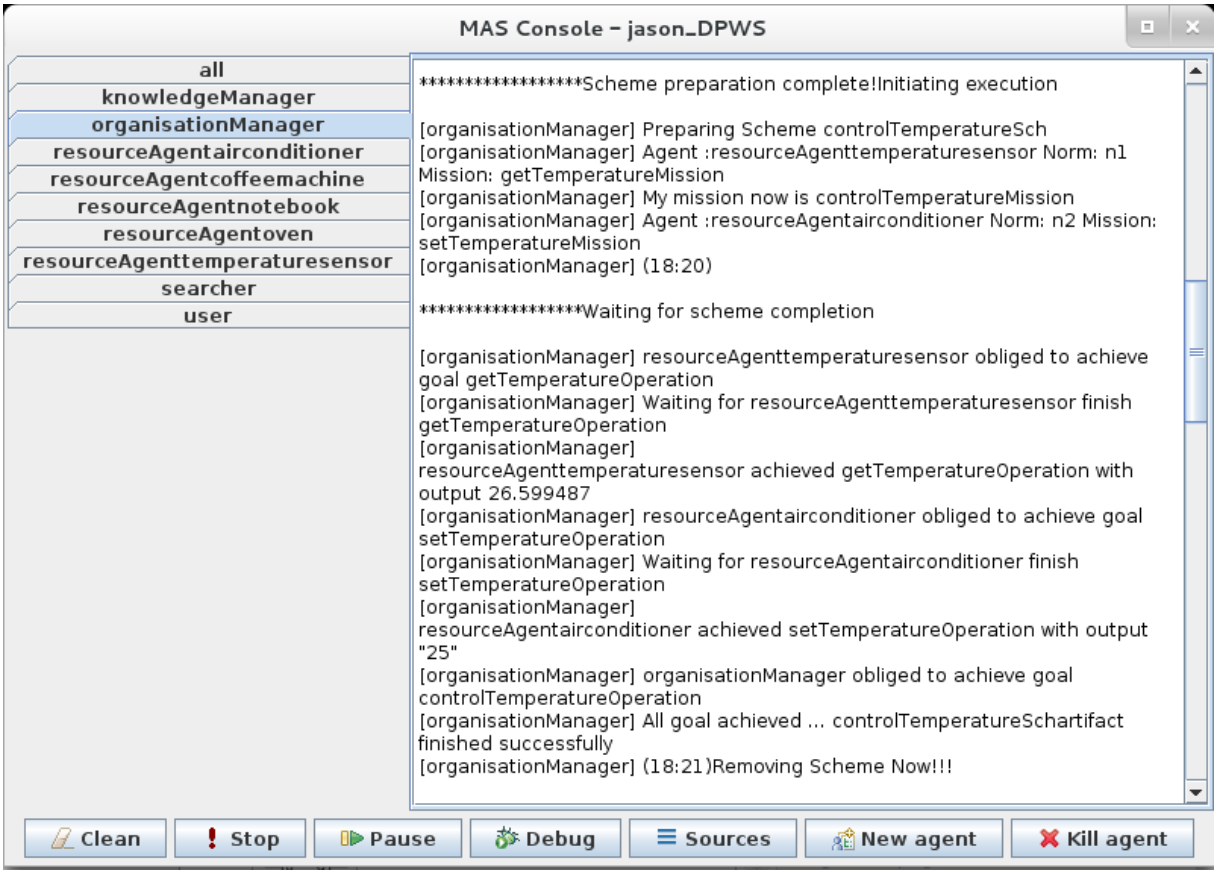

Figura 40: Mensagens do "Agente Gerenciador da Organização" durante a execução da tarefa "controlTemperatureSch".

6.3.4.2 Execução automatizada de tarefa agendada em um horário

Neste tipo de cenário uma dada tarefa do sistema deve ser executada em um determinado dia ou horário. No estudo de caso, a tarefa escolhida para representar a execução deste cenário é denominada "dailyMorningSch". A execução da tarefa “dailyMorningSch” ocorre somente quando os dispositivos forno elétrico, cafeteria e notebook estão presentes e será executada diariamente em um determinado horário, no caso todo dia às 7 horas da manhã.

A descrição segundo o diagrama OML de como esta tarefa será realizada é dada na Figura 41.

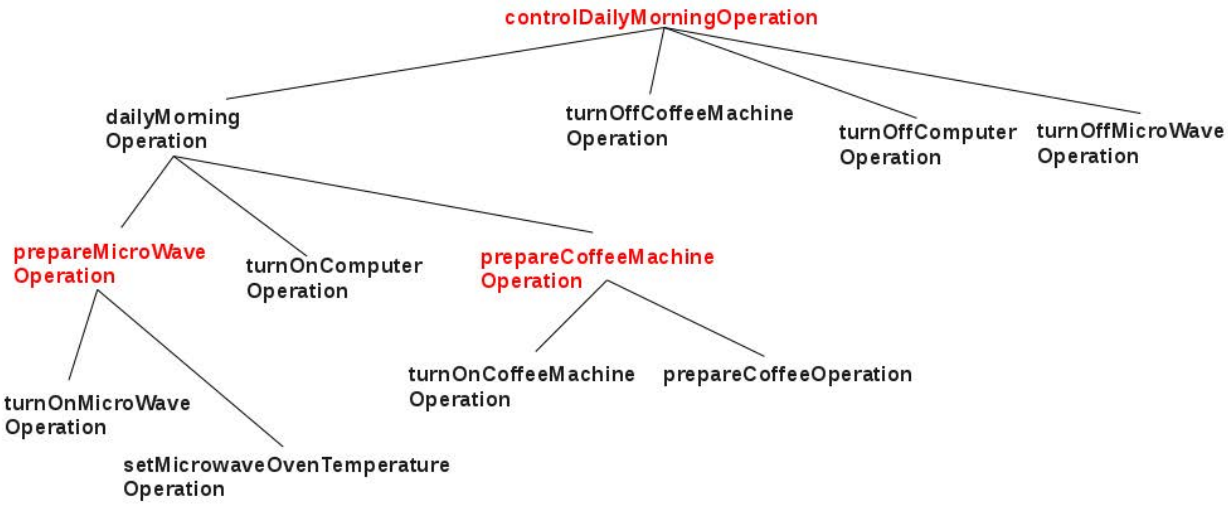

Figura 41: Descrição da tarefa "dailyMorningSch" segundo o diagrama OML.

No diagrama da Figura 41, os objetivos destacados em vermelho, representam pontos de sincronismo e correspondem a objetivos que devem ser atingidos pelo Agente Gerenciador da Organização. Isso significa que se algum objetivo não for atingido pelos Agentes de Recursos, ou mesmo for notificada uma falha durante sua execução, o Agente Gerenciador da Organização pode abortar a execução dessa tarefa.

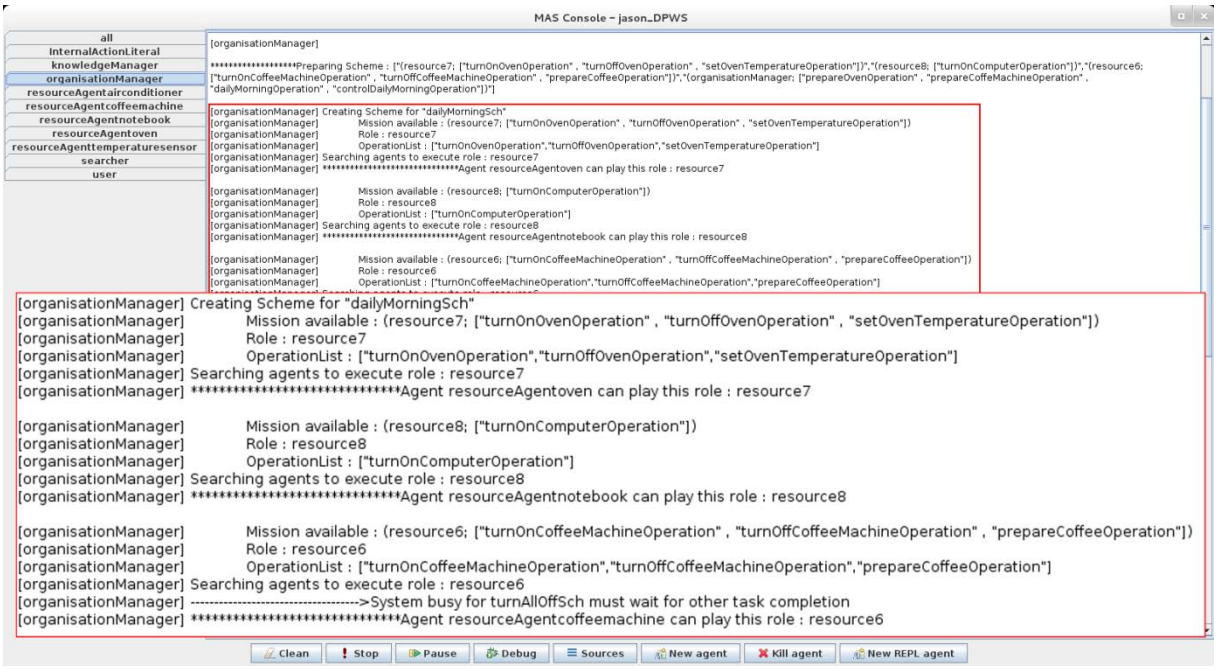

Figura 42: Mensagens do "Agente Gerenciador da Organização" durante a preparação da tarefa dailyMorningSch".

Da mesma forma que ocorre a execução das demais tarefas, existe uma fase de preparação onde, neste caso, o Agente Gerenciador da Organização deve encontrar os

Agentes de Recursos que gerenciam dispositivos do tipo notebook, forno elétrico e a cafeteira (Figura 42). Após realizada a fase de preparação da tarefa, a mesma entra em fase de execução.

A Figura 43 apresenta a ordem da execução das operações realizadas por cada um dos Agentes de Recursos. Destaca-se que a execução das operações realizadas por cada um dos agentes segue a ordem do diagrama OML (Figura 41).

Como saída da execução das operações realizadas pelos dispositivos observa-se o valor de 280, que corresponde ao valor de ajuste de temperatura do forno elétrico e, para operações de ligar ou desligar, o output é “false” informando que não ocorreu nenhum erro.

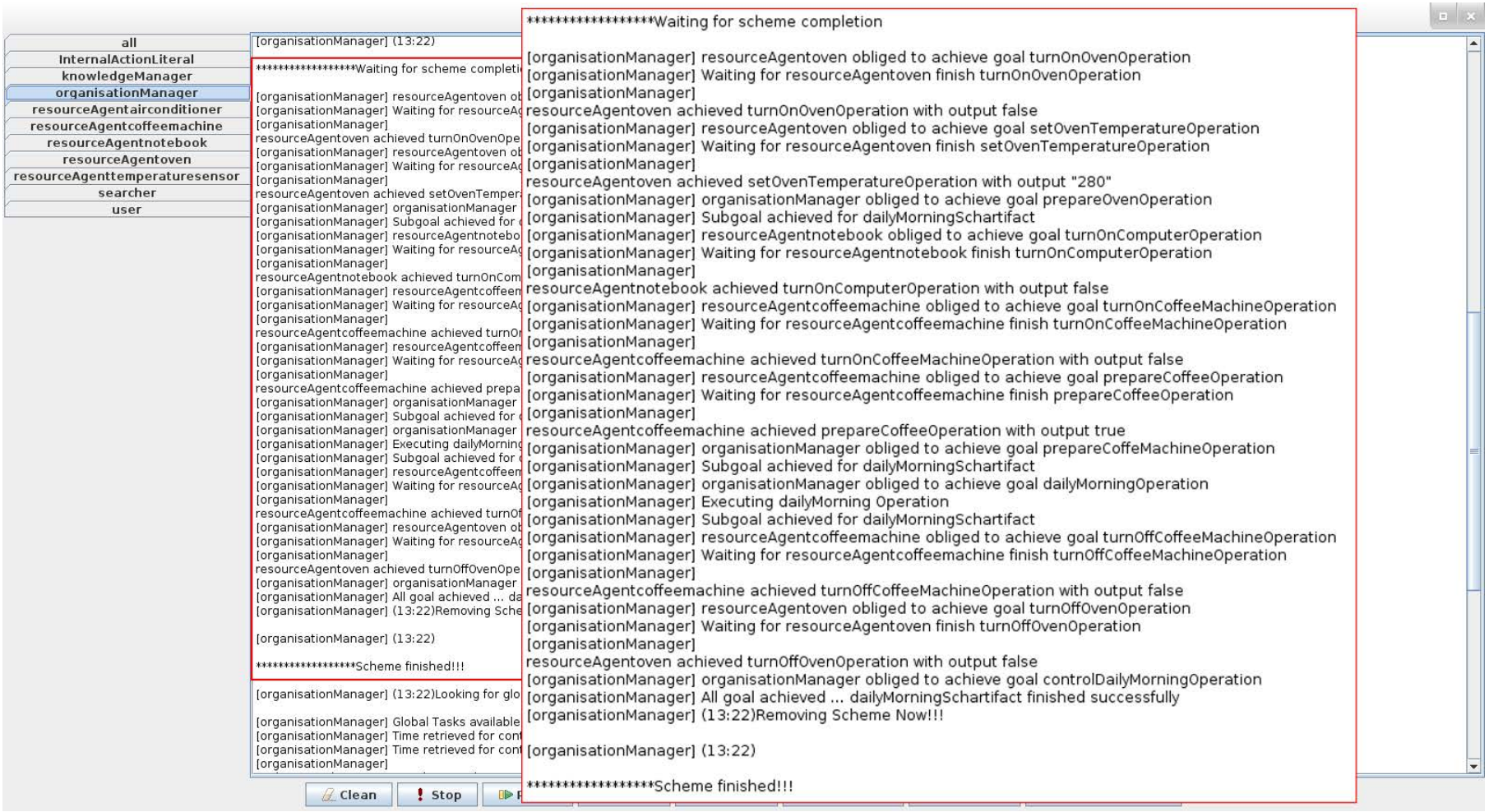

Figura 43: Mensagens do "Agente Gerenciador da Organização" durante a execução da tarefa "dailyMorningSch".

6.3.4.3. Execução de tarefa mediante requisição do usuário

A tarefa “turnAllOffSch” é executada após uma requisição do usuário da casa. Isso ocorre porque o contexto de identidade é definido como “homeuser” (Figura 30). Dessa forma, a tarefa será disparada quando o Agente de Usuários receber uma requisição do tipo HTTP (Figura 44).

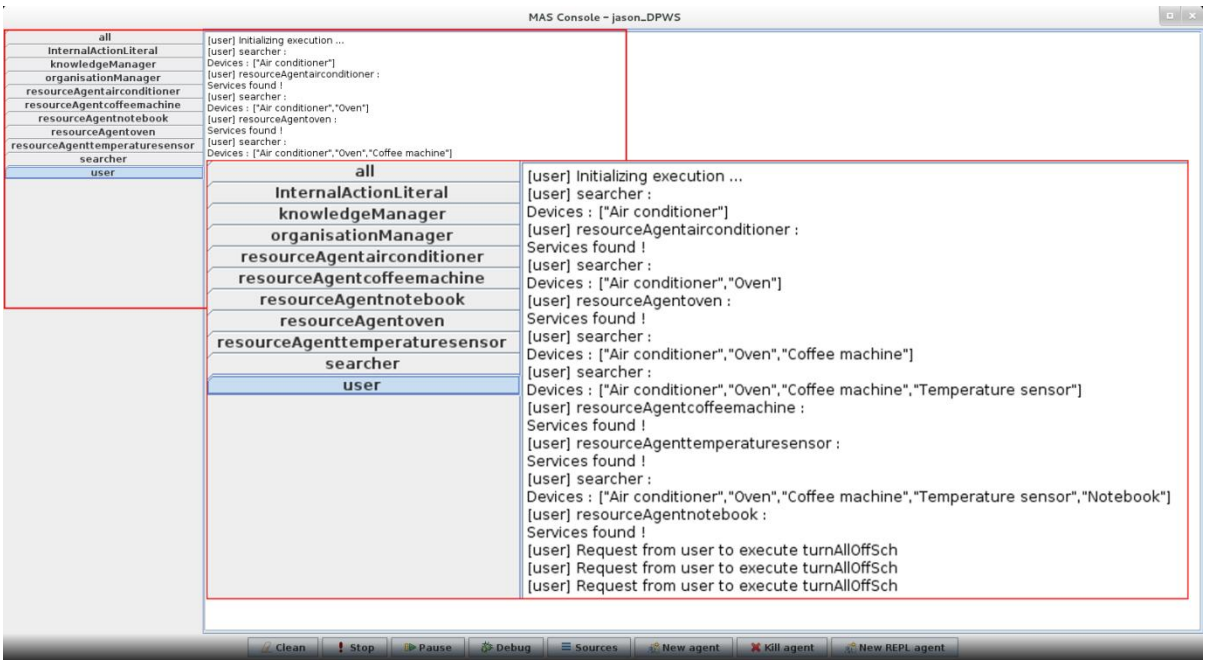

Figura 44: Mensagens do "Agente de Usuários" após a requisição de execução da tarefa “TurnAllOffSch”.

A figura 44 apresenta as mensagens do Agente de Usuários, entre as mensagens apresentadas destacam-se os dispositivos e serviços detectados até o momento, ao final observa-se a recepção da requisição do usuário para executar a tarefa "turnAllOffSch".

A figura 45 apresenta a árvore de objetivos que devem ser atingidos pelos agentes para a tarefa em questão. No diagrama apresentado na figura 44, os objetivos "turnOffOperation" e "turnOffComputerOperation" serão executados pelos Agentes de Recursos e o objetivo "controlTurnOffOperation" será executado pelo Agente Gerenciador da Organização.

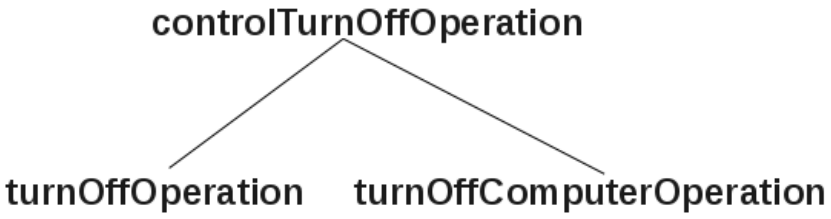

Figura 45: Descrição da tarefa organizacional "turnAllOffSch" segundo o diagrama OML.

A figura 46 demonstra a ordem da execução de cada uma destas tarefas.

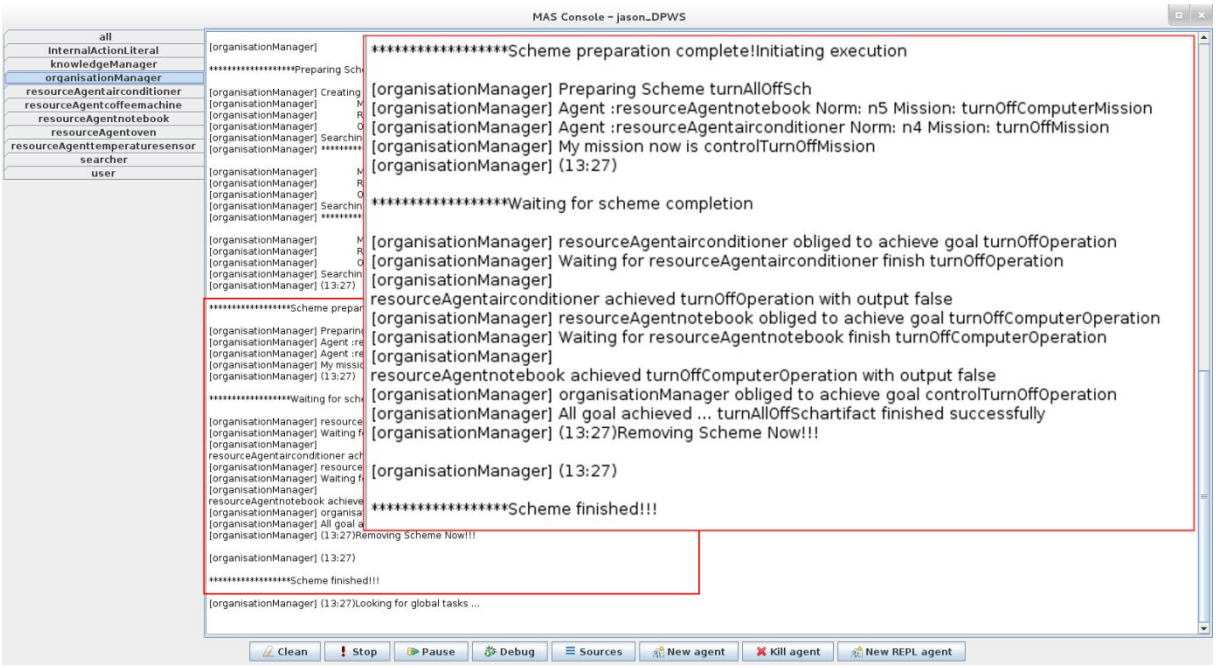

Figura 46: Mensagens do "Agente Gerenciador da Organização" durante a execução da tarefa “turnAllOffSch”.

Na figura 46 é destacada a ordem de execução das operações realizadas pelos agentes de recursos e pelo Agente Gerenciador da Organização. Durante a execução da tarefa os dispositivos que apresentam a operação "turnOffOperation" executam suas operações primeiro que os demais. Em seguida, as operações "turnOffComputerOperation" e "controlTurnOffOperation" são executadas nesta ordem.

6.4 Considerações e discussões

Nesse estudo de caso foi implementado um cenário de simulação em domótica, onde um conjunto de dispositivos simulando equipamentos de uso diário. Com base neste

problema, um protótipo de uma aplicação foi desenvolvido utilizando a arquitetura proposta no Capítulo 4.

Para essa aplicação foram desenvolvidos um conjunto inicial de agentes (Agente Pesquisador, Agente Gerenciador do Conhecimento, Agente de Usuário e Agente Gerenciador da Organização). Na Base de Conhecimentos foram criadas, na forma de ontologia, a descrição de três tarefas e seus contextos relacionados.

Esta aplicação foi submetida a testes baseados na descrição de cada um dos cenários apresentados no capítulo 4. Entre os testes realizados, verificou-se que a arquitetura apresentou a capacidade de identificação automática de novos dispositivos, fazendo uso da especificação de serviços DPWS e das informações semânticas destes recursos, utilizando a ontologia da Base de Conhecimentos.

Outro teste realizado foi a possibilidade da execução de tarefas que envolvam mais de um dispositivo. Dessa forma, foram utilizados os conceitos da programação orientada à organização, apresentada no capítulo 2, aliado ao uso de ontologias que permitiu a execução ordenada dos dispositivos durante a realização de tarefas para o usuário.

Por fim, o último teste realizado consistiu na execução de tarefas segundo a requisição de usuários, da mesma forma que apresentado anteriormente, a arquitetura também pôde realizar corretamente a tarefa.

É importante comentar que apesar dos resultados da implementação dos cenários forem positivos, nenhum parâmetro de eficiência foi medido durante a execução da arquitetura. Outra consideração a ser destacada está relacionada ao paralelismo. A implementação apresentada não permite a execução de tarefas de maior complexidade simultâneas. Isso significa que até que a organização de Agentes de Recursos não termine a execução da tarefa atual, nenhuma outra tarefa pode ser iniciada, ficando esta em fila para execução.

O principal objetivo desse estudo de caso foi alcançado mostrando como a arquitetura, que utiliza conceitos SOA, Sistemas Multiagentes e Ontologias, pode ser utilizada para desenvolver aplicações mais complexas que busquem, identificar automaticamente dispositivos presentes no ambiente da IoT e automatizar a execução de tarefas cotidianas de um usuário segundo a identificação de contextos.

7 Considerações Finais

Neste trabalho foi proposta uma arquitetura com base em SMA, SOA e tecnologias da Web Semântica para automatizar a integração e gerenciamento de dispositivos inteligentes em ambientes da Internet das coisas. Juntas, essas tecnologias podem ser usadas para criar uma infraestrutura de software para suportar o desenvolvimento de aplicações baseadas em ambientes de Internet das coisas, permitindo a execução de tarefas de acordo com as necessidades de contexto ou do próprio usuário.

Um protótipo do sistema foi implementado e testado em ambiente simulado no contexto de automação residencial. Neste contexto o sistema demonstrou a capacidade de adaptação, incorporando novos recursos; e flexibilidade, apresentado diversas possibilidade para execução de tarefas, bastando apenas alterar o domínio dos conceitos presentes na Base de Conhecimentos.

Apesar de apresentar resultados positivos quanto à implementação, nenhum parâmetro de eficiência foi medido neste trabalho. Dessa forma, a aplicação de tais conceitos em áreas crítica como engenharia de controle ou área afim, não pode ser totalmente determinada.

No desenvolvimento futuro deste trabalho os seguintes aspectos poderão ser considerados:

- Análise do consumo de memória, carga na rede e tempo de resposta.
- Adoção de outras tecnologias de interoperabilidade como serviços web ou REST.
- Implementação e teste dos estudos de caso em dispositivos físicos reais.
- Melhoria na descrição semântica dos dispositivos e reutilização de ontologias relacionadas, como por exemplo, a SSN (Semantic Sensor Network)
- Implementação da arquitetura proposta em outras tecnologias e frameworks de sistemas multiagentes, como por exemplo, Jadex ou Jade.
- Implementação de uma interface de usuário que possibilite a adição de novas tarefas de forma mais intuitiva.
- Melhoria na descrição semântica das regras da organização
- Adição de mecanismos de controle que permitam ao sistema executar mais de uma tarefa organizacional simultaneamente.

REFERÊNCIAS

ABOWD, G. D.; DEY, A.K.; BROWN, P. J.; DAVIES, N.; SMITH, M.; STEGGLES, P. Towards a better understanding of context and context-awareness. In: HUC '99. **Proc. 1st international symposium on Handheld and Ubiquitous Computing**, 1999, London, UK: Springer-Verlag, pp. 304–307.

ALOULOU, H.; MOKHTARI, M.; TIBERGHIEN, T.; BISWAS, J.; PHUA, C.; KENNETH, L.; JIN H.; YAP, P. Deployment of assistive living technology in a nursing home environment: methods and lessons learned. **BMC medical informatics and decision making**, v. 13, p. 42, jan. 2013.

ANGULO-LOPEZ, P.; JIMÉNEZ-PÉREZ, G. Collaborative Agents Framework for the Internet of Things. Intelligent Environments (Workshops), 2012. **Proceedings of the 8th International Conference on Intelligent Environments**, 2012: IOS Press, pp. 191-199

ATZORI L., IERA A., MORABITO G. The internet of things: A survey. **Computer Networks**, v. 54, pp. 2787-2805, 2010.

AZIZ, Z.; ANUMBA, C.; RUIKAR, D.; CARRILLO, P.; BOUCHLAGHEM, D. Semantic web based services for intelligent mobile construction collaboration, **Electronic Journal of Information Technology in Construction**, v. 9, p. 367-379, abr, 2004.

BAGNASCO, A.; CIPOLLA, D.; OCCHIPINTI, D.; PREZIOSI, A.; SCAPOLLA, A. M. Application of web services to heterogeneous networks of small devices. **WSEAS Transactions on Information Science and Applications**, v. 3, 5, p. 940-945, mai. 2006.

BATRES R., WEST. M., LEAL D., PRICE D., MASAKIA K., SHIMADA Y., FUCHINO T., NAKA Y. An Upper Ontology based on ISO 15926. **Computers & Chemical Engineering**, v. 31, n. 5-6, p. 519–534, 2007.

BENAZZOUZ Y.; MUNILLA, C. ; GUNALP, O. ; GALLISSOT, M. ; GURGEN, L. Sharing user IoT devices in the cloud. In: IEEE WORLD FORUM ON INTERNET OF THINGS (WF-IOT), 2014, Seoul. **IEEE World Forum on Internet of Things (WF-IOT)**. [s.l.]: [s.n.], 2014. p. 373 - 374.

BERNERS-LEE T.; HENDLER J.; LASSILA O. The Semantic Web. **Scientific American**, v. 284, 5, p. 29-37, mai. 2001.

BORDINI, R., HUBNER, J. F., VIEIRA, R. Jason and the Golden Fleece of Agent-Oriented Programming. In:_____. **Multi-Agent Programming.** Springer US, 2005. Cap. 1, p. 3-37.

BORDINI, R. H., HUBNER, J. F., and WOOLDRIDGE, M. **Programming Multi-Agent Systems in AgentSpeak Using Jason**. ed. John Wiley and Sons Ltd, 2007.

BOYLE, D. E.; YATES, D. C.; YEATMAN, E. M. Urban Sensor Data Streams: London 2013. **IEEE Internet Computing**, v. 17, n. 6, p. 12–20, nov. 2013.

BRAZIER, F. M. T.; KEPHART, J. O.; PARUNAK, H. V. D.; HUHNS, M. N. Agents and service-oriented computing for autonomic computing: A research agenda. **Internet Computing, IEEE**, 13, n. 3, 2009. 82–87.

CANDIDO, G.; JAMMES, F.; de Oliveira, J.B. ; COLOMBO, A.W. SOA at device level in the industrial domain: Assessment of OPC UA and DPWS specifications. In: IEEE INTERNATIONAL CONFERENCE ON INDUSTRIAL INFORMATICS (INDIN), 8., 2010, Osaka. **Procedings ....** [s.l.]: [s.n.], 2010. p. 598 - 603.

CHO, Hyuntae; KYUNG, Chong-min; BAEK, Yunju. Energy-efficient and fast collection method for smart sensor monitoring systems. In: ICACCI, 1., 2013, Mysore. Proceedings of ICACCI. 2013. [s.i], 2013. p. 1440 – 1445

COMPTON, M.; BARNAGHI, P.; BERMUDEZ, L.; GARCIA-CASTRO, R.; CORCHO, O.; COX, S.; GRAYBEAL, J.; HAUSWIRTH, M.; HENSON, C.; HERZOG, A.; HUANG, V.; JANOWICCZ, K.; KELSEY, W. D.; LE PHUOC, D.; LEFORT, L.; LEGGIERI, M.; NEUHAUS, H.; NIKOLOV, A.; PAGE, K.; PASSANT, A.; SHETH, A.; TAYLOR K. The SSN ontology of the W3C semantic sensor network incubator group, **Web Semantics: Science, Services and Agents on the World Wide Web**, v. 17, p. 25–32, dez. 2012.

CUCINOTTA, T. ; MANCINA, A.; ANASTASI, G. F.; LIPARI, G.; MANGERUCA, L.; CHECCOZZO, R. A Real-Time Service-Oriented Architecture for Industrial Automation. **IEEE Transactions on Industrial Informatics**, v. 5, n. 3, p. 267–277, 2009.

DIAS, R. A.; REGIS, A. Integrated Manufacturing Management using Internet of Things**. International Journal of Computer Applications,** v. 51, 11, p. 20–25, 2012.

DOHNDORF, O; KRUGER, J.; KRUMM, H.; FIEHE, C.; LITVINA, A.; LUCK, I.; STEWING, F. Towards the Web of Things: Using DPWS to bridge isolated OSGi platforms. In: IEEE INTERNATIONAL CONFERENCE ON PERVASIVE COMPUTING AND COMMUNICATIONS WORKSHOPS (PERCOM WORKSHOPS), 8., 2010, Mannheim. **Procedings ...** [s.l.]: [s.n.], 2010. p. 720 - 725.

FINN, T., FRITZSON, R.; MCKAY, D.; MCENTIRE, R. KQML as an Agent Communication Language. **The Proceedings of the Third International Conference on Information and Knowledge Management**, ACM Press, 1994.

FUENTES-FERNÁNDEZ, R.; GUIJARRO, M.; PAJARES, G. A multi-agent system architecture for sensor networks. In: ALKHATEEB, F.; MAGHAYREH, E. A.; DOUSH, I. A. **Multi-Agent Systems - Modeling, Control, Programming, Simulations and Applications**. InTech, 2011. Cap. 2. p. 23-40.

GASSER, L. **Organizations in multi-agent systems**. In Pre-Proceeding of the 10th European Worshop on Modeling Autonomous Agents in a Multi-Agent World (MAAMAW'2001),Annecy, 2001.

GUBBI, J.; BUYYA, R.; MARUSIC, S.; PALANISWAMI, M. Internet of Things (IoT): A vision, architectural elements, and future directions. **Future Generation Computer Systems**, v. 29, n. 7, p. 1645–1660, 2013.

GUILLEMIN P., FRIESS P. **Internet of Things Strategic Research Roadmap**, 2009. Disponível em <http://www.internet-of-things-research.eu/pdf/IoT_Cluster_Strategic_Research_Agenda_2011.pdf>. Acesso em: 5 jun. 2015

HAN, S.; LEE, G.; CRESPI, N. DPWSim: A simulation toolkit for IoT applications using devices profile for web services. 2014 In: IEEE WORLD FORUM ON INTERNET OF THINGS (WF-IOT), Seoul. **IEEE World Forum on Internet of Things (WF-IOT)**. [s.l.]: [s.n.], p. 544–547.

HAN, S. N.; LEE, G. M.; CRESPI, N. Semantic Context-Aware Service Composition for Building Automation System. **IEEE Transactions on Industrial Informatics**, v. 10, n. 1, p. 752–761, fev. 2014.

HÜBNER J., SICHMAN J, BOISSIER, O. Moise+: towards a structural, functional, and deontic model for mas organization. AAMAS, 2002. **Proceedings of the first international joint conference on Autonomous agents and Multi-Agent Systems**, p. 501-502. ACM Press, 2002.

HÜBNER J., SICHMAN, J., BOISSIER, O. Using the Moise+ for a Cooperative Framework of MAS Reorganisation. **Advances in Artificial Intelligence–SBIA,** 2004, 481-517

HÜBNER, J., SICHMAN, J., BOISSIER, O. **S-Moise+**: A middleware for developing organised multi-agent systems. In Proceedings of the International workshop on organizations in multi-agent systems, from organizations to organization oriented programming in MAS. LNCS (Vol. 3913, p. 64–78), 2006.

HÜBNER, J., BOISSIER; O., KITIO, R.; and RICCI A. Instrumenting multi-agent organisations with organisational artifacts and agents: "giving the organisational power back to the agents". **Journal of Autonomous Agents and Multi-Agent Systems**, v. 20, 3, p.369-400, 2010.

IZAGUIRRE, J. G.; LOBOV, A.; LASTRA, J. L. M. OPC-UA and DPWS Interoperability for Factory Floor Monitoring using Complex Event Processing. In: IEEE INTERNATIONAL CONFERENCE ON INDUSTRIAL INFORMATICS (INDIN), 9., 2011, Osaka. **Procedings ....** [s.l.]: [s.n.], p. 205–210.

JASPER, R., USCHOLD, M. A Framework for Understanding and Classifying Ontology Applications. A Framework for Understanding and Classifying Ontology Applications. In: WORKSHOP ON KNOWLEDGE ACQUISITION MODELING AND MANAGEMENT KAW 99, 20., 1999, Alberta. **Proceedings of the IJCAI-99 ontology workshop**. [s.l.]: [s.n.], 1999. p. 1 - 20.

KAKANAKOV, N.; SPASOV, G. Adaptation of Web service architecture in distributed embedded systems. In: INTERNATIONAL CONFERENCE ON COMPUTER SYSTEMS AND TECHNOLOGIES, 1., 2005, Varna. **Procedings ....** [s.l.]: [s.n.], 2005. v. 10, p. 1 - 6.

KARNOUSKOS, S.; TARIQ, M. M. J. Using multi-agent systems to simulate dynamic infrastructures populated with large numbers of web service enabled devices. In: INTERNATIONAL SYMPOSIUM ON AUTONOMOUS DECENTRALIZED SYSTEMS, 2009, Atenas. **Procedings ....** [s.l.]: [s.n.], 2009. v. 10, p. 1 - 7.

KOSOVAC, B., **A Framework for Managing Information from heterogeneous, Distributed, and Autonomous Sources in the Architecture, Engineering, Construction, and Facilities Management Domain**, PhD Thesis, Dept. of Civil Engineering, University of British Columbia, Canada, 2007.

KUKA, C.; NICKLAS, D. Enriching sensor data processing with quality semantics. In: IEEE INTERNATIONAL CONFERENCE ON PERVASIVE COMPUTING AND COMMUNICATIONS WORKSHOPS (PERCOM WORKSHOPS), 2014, Budapeste. **Procedings ....** [s.l.]: [s.n.], 2014. p. 437 - 442.

LEONG, P.; LIMING, L. Multiagent Web for the Internet of Things. In: INTERNATIONAL CONFERENCE ON INFORMATION SCIENCE AND APPLICATIONS (ICISA), 2014, Seoul. **Procedings ....** [s.l.]: [s.n.], 2014. p. 1 - 4.

MCCABE, F.; NEWCOMER, E.; CHAMPION, M.; FERRIS, C.; ORCHARD, D. **Web Services Architecture**, W3C Working Group Note, 2004. Disponíel em <http://www.w3.org/TRws-arch>. Acesso em: 20 mai. 2015.

MCARTHUR, S. D. J.; DAVIDSON, E. M.; CATTERSON, V. M.; DIMEAS, A. L.; HATZIARGYRIOU, N. D.; PONCI, F.; FUNABASHI, T. **Multi-Agent Systems for Power Engineering Applications - Part I**: Concepts, Approaches, and Technical Challenges, 22, n. 4, 2007. 1743-1752.

MOELLER, R.; SLEMAN, A. Wireless networking services for implementation of ambient intelligence at home. In: INTERNATIONAL CARIBBEAN CONFERENCE ON DEVICES, CIRCUITS AND SYSTEMS, 7., 2008, Cancun. **Procedings ...** [s.l.]: [s.n.], 2008. p. 1 - 4.

MURAR, M.; BRAD, S. Monitoring and controlling of smart equipments using Android compatible devices towards IoT applications and services in manufacturing industry. In: IEEE INTERNATIONAL CONFERENCE ON AUTOMATION, QUALITY AND TESTING, ROBOTICS, 2014, Cluj-napoca. **Procedings ...** [s.l.]: [s.n.], 2014. p. 1 - 5.

PAPAZOGLOU, M. P. Service-oriented computing: concepts, characteristics and directions. In: INTERNATIONAL CONFERENCE ON WEB INFORMATION SYSTEMS ENGINEERING, 1., 2003, San Diego. **Proceedings ...** [s.l.]: [s.n.], 2003. p. 3 - 12.

PAUTASSO, C.; ZIMMERMANN, O.; LEYMANN, F. Restful web services vs. "big"' web services: making the right architectural decision. In: INTERNATIONAL CONFERENCE ON WORLD WIDE WEB, 17., 2008, Pequim. **Proceedings ...** Nova York: Acm, 2008. p. 805 - 814.

PERERA, C.; ZASLAVSKY, A.; CHRISTEN, P.; GEORGAKOPOULOS, D. Context Aware Computing for The Internet of Things: A Survey. **IEEE Comunications**, v. 16, n. 1, p. 414–454, 2014.

PUJOLLE, G. An Autonomic-oriented Architecture for the Internet of Things. In: IEEE JOHN VINCENT ATANASOFF 2006 INTERNATIONAL SYMPOSIUM ON MODERN COMPUTING, 1., 2006, Sofia. **Proceedings ...** [s.l.]: [s.n.], 2006. p. 163 - 168.

PYNADATH, D. V., TAMBE M. An automated teamwork infrastructure for heterogeneous software agents and humans. **Autonomous Agents and Multi-Agent Systems**, vol. 7, n. 1-2, p. 71–100, 2003

QUEIROZ, J. F. P. **UMA ARQUITETURA MULTIAGENTE PARA SISTEMAS DE SUPERVISÃO E CONTROLE DE PROCESSOS**. 2013. 165 f. Dissertação (Mestrado) - Curso de Ciência da Computação, ", Instituto de Biociências, Letras e Ciências Exatas, – Universidade Estadual Paulista "júlio de Mesquita Filho", São José do Rio Preto, 2013.

RAO, A. S.; GEORGEFF, M. P. Modelling rational agents within BDI-architecture. In: INTERNATIONAL CONFERENCE ON PRINCIPLES OF REPRESENTATION AND REASONING, 2., 1991, San Mateo. **Proceedings …**, [s.l.]: [s.n.], 1991.

RAO, A. S. **AgentSpeak(L)**: BDI agents speak out in a logical computable language. In: WORKSHOP ON MODELLING AUTONOMOUS AGENTS IN A MULTI-AGENT

WORLD (MAAMAW'96), 7., Eindhoven, 1996, **Proceedings …**, [s.l.]:Springer-Verlag, 1996. p. 22 – 25.

RICCI A., PIUNTI, M., ACAY L. D., BORDINI R., HUBNER J., and DASTANI M. Integrating artifact-based environments with heterogeneous agent-programming platforms. In: INTERNATIONAL CONFERENCE ON AGENTS AND MULTI AGENTS SYSTEMS (AAMAS08), 7., Estoril, 2008. **Proceedings…**, [s.l.]: [s.n.], 2008. p. 225 – 232.

RICCI, A.; PIUNTI, M.; VIROLI, M. Environment Programming in Multi-Agent Systems – An Artifact-Based Perspective. **Autonomous Agents and Multi-Agent Systems**, v. 23 , n. 2, p. 158-192, 2011.

ROHOKALE, V. A.; PRASAD, N. R.; PRASAD, R. A cooperative Internet of Things (IoT) for rural healthcare monitoring and control. In: INTERNATIONAL CONFERENCE ON WIRELESS COMMUNICATIONS, VEHICULAR TECHNOLOGY, INFORMATION THEORY AND AEROSPACE & ELECTRONIC SYSTEMS TECHNOLOGY, 2, Le Royal Meridian Chennai, 2011. **Proceedings of Wireless Vitae 2011**, [s.l.]: [s.n.], 2011.p. 75 - 80.

RUTA, M.; SCIOSCIA, F.; DIi SCIASCIO, E.; ROTONDI, D.; PICCIONE, S. Semantic-Based Knowledge Dissemination and Extraction in Smart Environments. In: INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION NETWORKING AND APPLICATIONS WORKSHOPS, 27, Barcelona, 2013, **Proceedings…**, [s.l.]: [s.n.], 2013. p. 1289–1294.

RUSSELL, S.; NORVIG, P. **Inteligência artificial.** Rio de Janeiro, RJ: Elsevier, 2004.

SAMARAS, I. K.; HASSAPIS, G. D.; GIALELIS, J. V. A Modified DPWS Protocol Stack for 6LoWPAN-Based Wireless Sensor Networks. **IEEE Transactions on Industrial Informatics**, v. 9, n. 1, p. 209–217, 2013.

SHEN, W.; HAO, Q.; MAK, H.; NEELAMKAVIL, J.; XIE, H.; DICKINSON, J. ; THOMAS, R.; PARDASANI, A.; XUE, H. Systems integration and collaboration in architecture, engineering, construction, and facilities management**:** A review. **Advanced Engineering Informatics**, v. 24, n. 2, p. 196–207, 2010.

SHOHAM Y. Agent-oriented programming. **Artificial Intelligence**, v. 60, p.51–92, 1993.

SOLOMON, B.; IONESCU, D. Towards a Real-Time Reference Architecture for Autonomic Systems. In: INTERNATIONAL WORKSHOP ON SOFTWARE ENGINEERING FOR ADAPTIVE AND SELF-MANAGING SYSTEMS, Zurique, 7., 2007. **Proceedings…**, [s.l.]: [s.n.], 2007. p. 10–20.

SONG, Z.; CÁRDENAS, A.; MASUOKA, R. Semantic middleware for the Internet of Things. In: INTERNET OF THINGS (IOT), Tokyo, 2010. **Proceedings…**, [s.l.]: [s.n.], 2010. p. 1-8.

STAVROPOULOS, T. G.; GOTTIS, K.; VRAKAS, D.; VLAHAVAS, I. aWESoME: A web service middleware for ambient intelligence. **Expert Systems With Applications**, v. 40, n. 11, p. 4380–4392, 2013.

STIRBU, V. Towards a RESTful Plug and Play Experience in the Web of Things. In: IEEE INTERNATIONAL CONFERENCE ON SEMANTIC COMPUTING, Santa Clara, 2008. **Proceedings…**, [s.l.]: [s.n.], 2008, p. 512–517.

SUCIC, S.; BONY, B.; GUISE, L. Standards-compliant event-driven SOA for semantic-enabled smart grid automation: Evaluating IEC 61850 and DPWS integration. In: IEEE INTERNATIONAL CONFERENCE ON INDUSTRIAL TECHNOLOGY, Atenas, 2012. **Proceedings…**, [s.l.]: [s.n.], 2012, p. 403–408.

SUNDMAEKER H., GUILLEMIN P., FRIESS P. , and WOELFFLE S. , **Vision and challenges for realising the internet of things**, European Commission Information Society and Media, Tech. Rep. Disponível em <http://www.internet-of-things-research.eu/pdf/IoT Clusterbook March 2010.pdf> Acesso em: 10 out. 2014.

VAN ENGELEN, R. A.; GALLIVAN, K. A. The gSOAP toolkit for web services and peer-to-peer computing network. In: IEEE/ACM INTERNATIONAL SYMPOSIUM ON CLUSTER COMPUTING AND THE GRID, 2., Berlim, 2002. **Proceedings…**, [s.l.]: [s.n.], 2002. p.

WANG, X. Smart Sensor and Smart Sensor Networks**.** In: WORLD CONGRESS ON INTELLIGENT CONTROL, 5., Hangzhou , 2004. **Proceedings…**, [s.l.]: [s.n.], 2004. p. 3600–3606, 2004.

WOOLDRIDGE, M.; JENNINGS, N. R. Intelligent Agents: Theory and Practice. The Knowledge Engineering Review, v. 10, n. 2, p. 115-152, 1995.

Apêndice A.

Nesta seção serão apresentados a biblioteca de planos que nortearam a implementação de cada um dos agentes escritos em *AgentSpeak* para o interpretador Jason e a descrição de como estes planos são executados por cada um destes agentes.

A.1 Agente Pesquisador

A descrição detalhada de cada um dos planos bem como o contexto em que são executados é apresentada na tabela 5.

Tabela 5: Biblioteca de planos executados pelo "Agente Pesquisador".

Objetivo	Contexto	Descrição do plano
start	sempre executado	Plano de inicialização e adoção de papéis na organização.
setDPWSArtifact	constituição da organização	Plano executado uma vez que a organização está constituída. Este agente inicializa a camada de serviços, criando o artefato "SearchingBoard"
listen	dispositivo conectado ou não	Plano que corresponde à busca de dispositivos ou percepção de novos dispositivos conectados
newDeviceFound	dispositivo conectado	Plano executado quando um novo dispositivo, previamente desconhecido, está conectado na rede.
settingUpDevice	dispositivo conectado	Plano para obtenção dos metadados do dispositivo, solicitação da anotação semântica e criação de novos "Agentes de Recurso"
newDevice	sinal recebido do artefato	Sinal recebido pelo artefato da notificação de novos dispositivos na rede.

O agente desempenhando o papel Agente Pesquisador pode identificar novos dispositivos DPWS conectados com o auxílio das funcionalidades fornecidas pelo artefato "SearchingBoard" (vide seção 5.3). Basicamente, este agente percebe a conexão de um novo dispositivo a partir de um sinal enviado pelo artefato “SearchingBoard”. O sinal denominado "newDevice", quando recebido pelo agente, faz com este agente execute o plano associado a

este evento (percepção do sinal), onde é buscada as informações do novo dispositivo (nome, modelo e fabricante).

Os metadados obtidos são enviados para o agente desempenhado o papel de Agente Gerenciador de Conhecimentos, onde serão semanticamente anotados na base de conhecimentos (seção 5.3.1). O Agente Pesquisador ao identificar um novo dispositivo insere uma nova crença na lista com os nomes dos dispositivos conectados ao sistema.

A execução dos planos deste agente ocorre da seguinte forma, o agente deseja atingir o objetivo "start", como seu objetivo inicial. Isso cria uma intenção do plano de inicialização e adoção de papéis da organização. Uma vez que a inicialização da infraestrutura está concluída, este agente adota seu papel junto à organização e inicia a camada de serviços, a partir do *framework* JMEDS. Isso ocorre depois que o agente cria o artefato "SearchingBoard".

Após o sistema ser inicializado e a camada de serviços estar funcional, o agente executa continuamente os planos associados ao objetivo "listen", onde neste caso existem duas possibilidades: não há novos dispositivos ou novos dispositivos foram conectados à rede. Se nenhum dispositivo foi conectado, o agente volta a executar o plano "listen" novamente. Caso contrário, o agente executa o plano associado ao objetivo "newDeviceFound".

Quando o plano associado ao objetivo "newDeviceFound" é executado o agente envia uma solicitação ao Agente de Usuário, informando o nome dos novos dispositivos conectados e inicializa a busca semântica da descrição do dispositivo e a criação de novos Agentes de Recurso. Ao término deste plano, o agente volta novamente a executar a busca por dispositivos, executando o plano associado ao objetivo "listen".

Outra possibilidade para a detecção de novos dispositivos ocorre quando o artefato "SearchingBoard" envia um sinal "newDevice" para este agente. Neste caso após receber o sinal o agente executa o plano associando ao objetivo "newDeviceFound".

Por fim, este agente ainda informa o Agente de Usuário a ocorrência de possíveis erros na execução de cada um dos planos acima descritos.

A.2 Agente de Recursos

O agente desempenhando o papel de "Agente de Recursos", tem a função de gerenciar a execução de um dispositivo segundo uma determinada tarefa realizada pelo sistema. Neste caso, este agente apresenta planos e crenças associadas aos dados, ao estado e as operações

realizadas pelo dispositivo. Na tabela 6 é apresentado todos os planos que compõem a biblioteca de planos deste agente.

Tabela 6: Biblioteca de planos executados pelo "Agente de Recursos".

Objetivo	**Contexto**	**Descrição do plano**
device	Após sua criação e após receber do agente Pesquisador o nome do dispositivo a ser gerenciado	É o primeiro plano que este agente executa e tem por objetivo inicializar o agente.
role	É executado quando o "Agente Gerenciador da Organização" envia na forma de crença o papel que este agente deve assumir na organização.	Neste plano o "Agente de Recursos" entra na organização de agentes do sistema.
setArtifact	Sempre executado.	Inicializa o artefato "Device" que permitirá à este agente conectar-se aos dispositivos a partir da camada de serviços.
getServices	Executado se o artefato foi corretamente inicializado	Obtém os metadados de todos os serviços do dispositivo e solicita ao "Agente Gerenciador de Conhecimentos" a busca pela descrição semântica dos serviços do mesmo. Também informa ao "Agente Gerenciador da Organização" os novos recursos disponíveis na forma de serviços.
getOperations	Se um serviço foi encontrado no dispositivo	Para cada serviço do dispositivo, obtêm os metadados das suas operações. Solicita a busca pela descrição semântica.
createOperationGoals	Quando uma operação for encontrada.	Para cada operação encontrada, define o nome da operação como um objetivo para um novo plano.
addOperationPlan	Se o objetivo cujo nome corresponde à operação do dispositivo foi criado.	Um novo plano é criado e adicionado na base de planos do agente. Agora o agente pode executar a operação do dispositivo como se a mesma fosse um plano pré-definido.

executeOperation	Se um objetivo que corresponde ao nome de uma operação necessitar ser atingido.	Executa a operação do dispositivo.
operationParameter	Se a operação necessitar de parâmetros de entrada.	Recebe o valor dos parâmetros de entrada para execução da operação correspondente.
commitMission	Se o agente adotou um papel na organização e uma tarefa que envolva o dispositivo por ele gerenciado necessitar ser executada.	Faz com que o agente seja responsável pela obtenção dos objetivos internos à missão.
outputFromOperation	Após executada a operação e se houver saída.	Recebe o valor da saída produzida pela execução de uma operação na forma de uma crença.
operationFinished	Se a operação não retorna valor de saída mas já foi executada	Recebe um sinal informando o término da operação executada.

A execução dos planos deste agente ocorre após sua criação quando o Agente Pesquisador identifica e anota semanticamente um novo dispositivo na rede. Em seguida, o Agente Pesquisador envia uma crença contendo o nome do dispositivo para este agente. Este evento dispara a execução do primeiro plano do Agente de Recursos, cujo objetivo é denominado "device". Este plano depois de executado faz com que o agente inicie a conexão e identificação dos serviços e operações do dispositivo.

Para realizar a conexão com o dispositivo este agente criará um artefato. O artefato chamado "Device" possibilita ao agente conectar com os dispositivos físicos DPWS conectados na rede e consumir seus serviços.

No momento em que este agente se conecta com o dispositivo, ele recupera todos os serviços e operação expostos pelo dispositivo. Isso é realizado quando o agente executa em sequência os planos cujos objetivos que o agente deseja atingir são "getServices" e "getOperations" (vide descrição na tabela 5).

Depois que os serviços e as operações foram identificados, o agente executará o plano para atingir o objetivo "createOperationGoals", onde o agente criará um novo plano cujo objetivo apresenta o nome de uma das operações do dispositivo. Uma vez criado o novo plano, o agente o adiciona em sua base de planos ("addOperationPlan").

O exemplo a seguir demonstra como ocorre a execução destes dois planos. Suponha que ao ser criado pelo "Agente Pesquisador", o novo "Agente de Recursos" gerencia um dispositivo chamado sensor de temperatura, cuja operação é denominada "getTemperature". Ao executar o plano e atingir o objetivo "createOperationGoals", o agente terá criado um novo plano cujo objetivo é "getTemperature", que será utilizado durante a execução das tarefas da organização. Dessa forma, todas as operações recuperadas do dispositivo, são convertidas para uma representação na linguagem *AgentSpeak* e tornando-se novos planos, que são adicionados à biblioteca da planta de agente. Como consequência, este agente pode agora executar as operações expostas pelo dispositivo garantindo assim a adaptabilidade do SMA.

Ao terminar esta sequência de ações acima descritas, a execução dos planos deste agente será disparada somente quando o mesmo estiver envolvido em tarefas que envolvam os demais agentes da organização. Isso significa que os demais planos de sua biblioteca de planos estão relacionados à uma tarefas que foi previamente identificada na base de conhecimentos segundo os recursos necessários.

A.3 Agente Gerenciador do Conhecimento

O agente desempenhando o papel de "Agente Gerenciador de Conhecimentos" tem como função gerenciar os conhecimentos da aplicação descritos na forma de ontologias e fornecer uma interface para que os demais agentes deste sistema possam também utilizar este conhecimento. Além do gerenciamento das ontologias utilizadas este agente também permite que novos dados possam ser adicionados na forma de conceitos, permitindo renovação e adição de novos conceitos, garantindo assim maior flexibilidade ao sistema.

Os planos executados por este agente são dados na tabela 7.

Tabela 7: Biblioteca de planos executados pelo "Agente Gerenciador do Conhecimento".

Objetivo	**Contexto**	**Descrição do plano**
start	sempre executado	Plano que inicializa o agente
prepareKnowledgeBase	Depois que ocorreu a inicialização do agente	Inicializa a base de conhecimentos, carregando as ontologias em memória primária e traduzindo os conceitos da ontologia no arquivo XML de configuração do Moise. Depois de criado este arquivo, um

		notificação e enviada para o "Agente Gerenciador da Organização".
lookForAGlobalTask	Executado segundo a solicitação do "Agente Gerenciador da Organização"	Pesquisa na base de conhecimentos, quais são as tarefas (descritas na dimensão funcional da organização) disponíveis segundo os recursos conectados a rede.
getTaskInfo	Executado segundo a solicitação do "Agente Gerenciador da Organização"	Responde ao "Agente Gerenciador da Organização" como a tarefa deve ser executada. Isso significa que envia quais são os papeis, missões, objetivos e dispositivos associados à tarefa.
annotateDevice	Executado segundo a solicitação do "Agente Pesquisador"	Para execução deste plano o agente recebe os metadados do dispositivo e identifica na ontologia sua descrição. Uma vez identificado o agente cria um novo individuo na ontologia.
annotateServices	Executado segundo a solicitação do "Agente de Recursos"	Neste plano o agente realiza a notação semântica dos serviços uma vez que os mesmos foram identificados pelo "Agente de Recursos"
annotateOperations	Executado segundo a solicitação do "Agente de Recursos"	Neste plano realiza a notação semântica das operações do dispositivo.
closeKnowledgeBase	Executado periodicamente ou se algum erro ocorrer no sistema	Plano executado quando a ontologia for escrita em arquivo. Neste caso a base pode ser completamente fechada não permitindo mais nenhuma consulta, ou o conhecimento é escrito em arquivos, tornando a base temporariamente indisponível.
getAllDevicesAnnotated	Executado segundo a solicitação do "Agente Gerenciador da Organização"	Responde ao "Agente Gerenciador da Organização" quais são dispositivos que estão ou já foram conectados que já foram anotados.

A.4 Agente Gerenciador da Organização

Este papel tem por objetivo gerenciar a organização de agentes do sistema em suas três dimensões (estrutural, funcional e normativa). Este agente realiza esta função com o auxílio de dois artefatos pré-definidos na API ORA4MAS, chamados "GroupBoard" e

"SchemeBoard" (seção 5.3.2). Os planos deste agente incluem ações para a execução de esquemas funcionais e obtenção de objetivos da organização (Tabela 8).

Tabela 8: Biblioteca de planos executados pelo "Gerenciador da Organização".

Objetivo	Contexto	Descrição do plano
start	Executado quando a base de conhecimentos foi corretamente inicializada e os arquivos de configuração do *framework* Moise foram criados	Inicializa a infraestrutura da organização a partir da criação do artefato organizacional "GroupBoard".
lookForGlobalOperations	A infraestrutura da organização (artefatos ORA4MAS) foi corretamente inicializada.	Solicita ao "Agente Gerenciador de Conhecimentos" sob quais contexto e quais são as tarefas que podem ser executadas na organização.
create_scheme	Existem tarefas que podem ser executadas com os recursos conectados atuais foram encontradas.	Inicializa a preparação da execução de uma tarefa da organização criando um artefato "SchemeBoard" que controlará a execução da tarefa.
settingUpRoles	O artefato foi devidamente criado.	Encontra os papéis e missões que cada "Agente de Recurso" desempenhará durante a execução da tarefa.
settingUpResources	O papel foi encontrado	Localiza o agente que gerencia o recurso associado à missão da tarefa
settingUpAgent	O agente foi localizado	Envia o papel e a missão que o agente deve desempenhar e cumprir para a realização da tarefa.
settingSchemeParameters	Se uma dada missão apresenta alguma operação que necessite de parâmetros de entrada	Depois que consultado na ontologia, envia o valor do parâmetro para o agente.
lookForTaskInfo	O "Agente Gerenciador de Conhecimentos" enviou as tarefas disponíveis	Este plano quando executado apresenta ao agente as informações necessárias para execução da tarefa (papéis, missões e dispositivos).
getSpecification	Executado quando o "Agente Gerenciador da Organização" deseja consultar as especificações	Quando executado apresenta ao agente a especificação da organização, descrita no arquivo XML de configuração do Moise.

	da organização	
operationFinished	Notificação que a operação realizada por um dos "Agentes de recursos" finalizou a operação e alcançou seu objetivo.	Este plano é utilizado para notificar o término de uma operação executada por um "Agente de Recursos"
obligation	É executado quando um novo objetivo deve ser alcançado segundo a sequência de planos da tarefa da organização	Quando executado faz o gerenciamento e a coordenação dos objetivos que devem ser alcançados pelos "Agentes de Recursos" da organização.

A execução dos planos deste agente é iniciado logo após a criação e a inicialização da base de conhecimentos, executados pelo Agente Gerenciador de Conhecimentos. Uma vez que a base de conhecimentos foi inicializada, o arquivo XML de configuração do Moise também foi criado e dessa forma o "Agente Gerenciador da Organização" inicia a criação do artefato GroupBoard, onde as especificações da organização estão pré-definidas.

Após finalizar a execução do primeiro plano, o agente inicia a execução do plano cujo objetivo é denominado "lookForGlobalOperations", onde é feita a solicitação para o "Agente Gerenciador de Conhecimentos" informar todos os dispositivos conectados e detectados até momento.

A resposta à esta requisição será tratado no plano cujo objetivo é "lookForTaskInfo", neste caso, este agente recebe uma lista das possíveis tarefas que poderão ser executadas pela organização de agentes. Para cada tarefa encontrada na lista, o "Agente Gerenciador da Organização" faz os preparativos para a execução da tarefa, notificando os Agentes de Recursos que participarão, executando o plano associado ao objetivo "settingUpResources". Neste caso, o Agente Gerenciador da Organização deve identificar e informar ao agente participante, a missão (conjunto de planos) a ser executada e o nome do papel que este agente assumirá durante a execução da tarefa.

O controle da execução de tarefas é executada pelo artefato ORA4MAS denominado "SchemeBoard" que apresentará aos Agentes de Recursos a ordem e o momento correto de execução de cada umas das operações dos dispositivos (seção 2.2.4.3). Isso significa que toda tarefa executada pelo sistema deve antes estar previamente cadastrada na base de conhecimentos de acordo com as especificação utilizadado pelo *framework* Moise .

A.5 Agente de Usuário

Finalmente, o Agente de Usuário tem planos para imprimir mensagens de agentes e imprimir informações sobre o estado de um objetivo ou tarefa da organização. Além disso, este agente também pode receber as requisições de usuários do sistema. A biblioteca de planos deste agente é descrita na tabela 9.

Tabela 9: Biblioteca de planos executados pelo "Agente de Usuário".

Objetivo	Contexto	Descrição do plano
start	sempre executado.	Plano incial executado por este agente responsável por inicializar uma interface Web para recepção de mensagens HTTP.
print	Recepção de mensagens provindas de outros agentes do sistema.	Exibe a mensagem de um dos agentes na interface Web. Recebe como parâmetro a mensagem passada pelo agente.
registerGui	Um novo "Agente de recurso" foi criado.	Cadastra o novo agente de modo a receber mensagens deste agente
printError	Um erro ocorreu durante a execução de alguma tarefa.	Exibe uma mensagem de erro passada por outro agente.
userRequest	O usuário está solicitando a execução de alguma tarefa.	Envia para o "Agente Gerenciador da Organização" a tarefa a ser executada que foi solicitada pelo usuário.

A execução deste agente é iniciada com o plano “start” cujo objetivo é inicializar a interface de rede que receberá requisições do usuário via protocolo HTTP. Os demais planos deste agente serão executados em resposta às solicitações dos demais agentes.

Após o Agente Gerenciador da Organização identificar as tarefas globais do contexto, ele informa essas tarefas para o Agente de Usuário que recebe requisições HTTP, permitindo ao usuário interagir com o sistema. Toda vez que o usuário solicita uma tarefa, este agente pode informar o Agente Gerenciador da Organização para iniciar um novo objetivo organização.

Apêndice B.

Nessa seção são descritos os dispositivos, serviços e operações utilizadas durante a execução dos cenários do estudo de caso. A tabela 10 relaciona os serviços e operações de cada um destes dispositivos.

Tabela 10: Descrição dos dispositivos utilizados no estudo de caso.

Dispositivo	Serviço	Operação	Descrição
AirConditioner	TemperatureControl	SetTemperature	Ajusta a temperatura para um dado valor. Esta operação recebe como parâmetro de entrada o valor da temperatura
CoffeeMachine	PrepareCoffee	PrepareCoffee	Simula a preparação de um café para o usuário.
TemperatureSensor	GetTemperature	GetTemperature	Retorna o valor medido da temperatura ambiente.
Oven	ControlOvenTemperature	SetOvenTemperature	Ajusta a temperatura do forno. A um dado valor recebe como parâmetro o valor da temperatura.
Notebook	TurnComputerOnOff	TurnComputerOn	Liga o computador. Retorna verdadeiro se ocorreu algum erro no processo.
		TurnComputerOff	Desliga o computador. Retorna verdadeiro se ocorreu algum erro no processo.

Estes dispositivos foram implementados a partir da API do JMEDS e conectados a rede local e serviram de recursos para que os agentes da arquitetura pudessem consumir seus serviços e executar suas operações.

Apêndice C.

Arquivo de configuração utilizado pelo *framework* Moise, criado a partir da ontologia de aplicação desenvolvida para o estudo de caso. Trechos do arquivo que representam as dimensões estrutural, funcional e normativa da organização são descritos a seguir.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<organisational-specification xmlns="http://moise.sourceforge.net/os" id="csoHouse" os-version="0.8">
    <structural-specification>
        <role-definitions>
            <role id="cognitiveAgent"/>
            <role id="reactiveAgent"/>
            <role id="knowleageManager">    <extends role="cognitiveAgent"/> </role>
            <role id="organisationManager"> <extends role="cognitiveAgent"/> </role>
            <role id="resource"> <extends role="reactiveAgent"/> </role>
            <role id="resource1"> <extends role="resource"/> </role>
            <role id="resource10"> <extends role="resource"/> </role>
            <role id="resource2"> <extends role="resource"/> </role>
            <role id="resource3"> <extends role="resource"/> </role>
            <role id="resource4"> <extends role="resource"/> </role>
            <role id="resource5"> <extends role="resource"/> </role>
            <role id="resource6"> <extends role="resource"/> </role>
            <role id="resource7"> <extends role="resource"/> </role>
            <role id="resource8"> <extends role="resource"/> </role>
            <role id="resource9"> <extends role="resource"/> </role>
            <role id="searcher">  <extends role="reactiveAgent"/> </role>
            <role id="user"> <extends role="cognitiveAgent"/> </role>
        </role-definitions>
    </structural-specification>
```

A dimensão estrutural apresenta todos os papéis que os agentes podem adotar dentro da organização durante a execução de uma tarefa. Neste caso, os agentes desempenhando os papéis de Agente Pesquisador, Agente Gerenciador de Conhecimentos, Agente Gerenciador da Organização e Agente de Usuários, adotam o seu papel na inicialização do sistema. Já para que o sistema possa realizar tarefas de maneira flexível, os Agentes de Recurso criados durante a execução do sistema adotam papéis temporários, somente durante a execução da

tarea. Dessa forma, quando a tarefa e finalizada, o Agente de Recursos deixa o seu papel na organização.

Os papéis temporários dos Agentes de Recursos são utilizados para que os mesmos possam executar as tarefas especificadas na organização. A execução da tarefa dependerá do dispositivo gerenciado. Abaixo é apresentado um trecho da dimensão funcional da organização que descreve, a tarefa "turnAllOffSch" apresentando as missões e a ordem com que alguns objetivos devem ser atingidos pela organização.

```
<functional-specification>
<scheme id="turnAllOffSch">
        <goal id="controlTurnOffOperation">
        <plan operator="sequence">
                <goal id="turnOffOperation"/>
                <goal id="turnOffComputerOperation"/>
        </plan>
        </goal>
        <mission id="turnOffMission">
                <goal id="turnOffOperation"/>
        </mission>
        <mission id="turnOffComputerMission">
                <goal id="turnOffComputerOperation"/>
        </mission>
        <mission id="controlTurnOffMission">
                <goal id="controlTurnOffOperation"/>
        </mission>
</scheme>
</ functional-specification>
```

Os papéis que deverão se encarregar da execução de cada uma destas missões estão relacionados com a dimensão normativa. Abaixo é apresentado um trecho do mesmo arquivo que descreve a dimensão normativa associada à tarefa "turnAllOffSch"

```
<normative-specification>

 <norm id="n3" role="organisationManager" type="obligation" mission="controlTurnOffMission"/>
<norm id="n4" role="resource3" type="obligation" mission="turnOffMission"/>
<norm id="n5" role="resource10" type="obligation" mission="turnOffComputerMission"/>
```

</normative-specification>

É importante destacar que o uso da descrição da organização na forma de ontologias permite com que o sistema possa criar diferentes arquivos de configuração para o *framework* Moise e a infraestrutura ORA4MAS.