Programa Interdisciplinar de Pós-Graduação em

Computação Aplicada

Mestrado Acadêmico

Pedro Ricardo Oliveira

Fayol: Um Sistema Multi-agente de Gerência e Controle de Acesso à Plataforma Milos

São Leopoldo, 2014

Pedro Ricardo Oliveira

FAYOL: UM SISTEMA MULTI-AGENTE DE GERÊNCIA E CONTROLE DE ACESSO À PLATAFORMA MILOS

Dissertação apresentada como requisito parcial para a obtenção do título de Mestre pelo Programa de Pós-Graduação em Computação Aplicada da Universidade do Vale do Rio dos Sinos — UNISINOS

Coordenador:
Prof. Dr. Cristiano André da Costa

Orientador:
Prof. Dr. João Carlos Gluz

São Leopoldo
2014

FICHA CATALOGRÁFICA

O48f Oliveira, Pedro Ricardo
Fayol: um sistema multi-agente de gerência e controle de acesso à plataforma Milos / Pedro Ricardo Oliveira – 2014.
64f. : il.

Dissertação (mestrado) – Universidade do Vale do Rio dos Sinos, Programa de Pós-Graduação em Computação Aplicada, São Leopoldo, RS, 2014.
"Orientador: Prof. Dr. João Carlos Gluz"

1. Computação Aplicada. 2. Software - Desenvolvimento. 3. Objetos de aprendizagem. 4. Ontologias. 5. Autenticação I. Título.

CDU 004
004.4
004:37

Catalogação na fonte:

Mariana Dornelles Vargas – CRB 10/2145

Pedro Ricardo Oliveira

FAYOL: UM SISTEMA MULTI-AGENTE DE GERÊNCIA E CONTROLE DE ACESSO À PLATAFORMA MILOS

Dissertação apresentada como requisito parcial para a obtenção do título de Mestre pelo Programa Interdisciplinar de Pós-Graduação em Computação Aplicada da Universidade do Vale do Rio dos Sinos – UNISINOS.

Aprovado em: ___/____/____.

BANCA EXAMINADORA

_____

Prof. Dr. João Carlos Gluz – UNISINOS (orientador)

_____

_____

À minha esposa Adri, meus filhos Mônica, Pedro e Helena
pelo apoio na decisão de enfrentar a este desafio.
Aos meus pais Adão *(in memorian)* e Célia
por tudo que fizeram por mim ao longo de minha vida.
Aos meus irmãos pelo carinho e companheirismo.

AGRADECIMENTOS

Agradeço ao meu professor orientador pela paciência e pela ajuda na conclusão deste trabalho. Agradeço também aos colegas e professores do curso e, principalmente aos colegas de projeto, pelo companheirismo e pelas conversas animadas nas rodas de café dos intervalos de pesquisa. Sem vocês não teria conseguido superar este que foi, sem dúvida, meu maior desafio até então.

"*Depois de escalar uma montanha muito alta, descobrimos que há muitas outras montanhas por escalar.*"
(Nelson Mandela)

RESUMO

A utilização crescente de objetos de aprendizagem e, em consequência, de plataformas de conteúdos que realizam as funções de armazenagem, criação, modificação e consulta de forma gerenciada e controlada, cria a necessidade de um mecanismo de autorização e controle de acesso a estas plataformas. Inserido em uma plataforma de conteúdos específicos que suporta o ciclo de vida completo de objetos de aprendizagem, o presente trabalho tem por objetivo desenvolver uma ferramenta que irá prover os serviços de autenticação e autorização (controle de acesso) de forma integrada e compatível com as tecnologias relacionadas a ontologias, web semântica e agentes inteligentes de software, que compõem o ambiente operacional da plataforma. O trabalho procura explorar o potencial destas tecnologias e ferramentas, e sua efetividade na especificação e detalhamento de um modelo e mecanismo de autorização e controle de acesso. A análise do estado da arte mostra que a aplicação das tecnologias de sistemas multiagente e ontologias nas questões de autorização e controle de acesso é uma tendência de pesquisa importante, mas muito recente. Assim o trabalho pretende contribuir com o avanço dessas pesquisas. Ao propor um modelo ontológico completo para autenticação e controle de acesso, além de um mecanismo baseado em agentes, federado e com comunicação segura que implementa este modelo, a presente dissertação explora possibilidades ainda não consideradas nessa nova abordagem, mas presentes em mecanismos mais tradicionais de autenticação e autorização. A avaliação do modelo proposto e do mecanismo implementado foi realizada através de experimentos funcionais e de desempenho realizados em laboratório, seguindo a prática de avaliação dos mecanismos atuais de autenticação e autorização.

Palavras-chave: Autenticação. Autorização. Controle de Acesso. Agentes de Software. Engenharia de Software. Ontologias.

ABSTRACT

The increase use of learning objects and content platforms that perform the functions of storage, creation, modification and query of these objects on a managed and controlled manner, creates the need for mechanisms to control the access to these platforms. Inserted into a specific content platform that supports the complete life-cycle of learning objects, the present work aims to develop a tool that will provide authentication and authorization (access control) services, integrated and compatible with the technologies already in use on platform. The work makes use of ontologies, semantic web and intelligent software agents technologies that comprise the operating environment of the platform, looking to exploit the potential of these technologies and tools, and its effectiveness in the specification and detailing models and mechanisms for authentication and access control. The analysis of the state of the art shows that the application of multi-agent systems and ontologies technologies on authorization and access control questions is an important, but very recent, research trend. Thus, this work aims to contribute to the advancement of such research. This dissertation proposes an complete ontological model for authentication and access control plus an agent-based, federated and secure communication mechanism that implements this model. As a result, this dissertation explores possibilities not yet considered in this new approach, but present in more traditional mechanismos of authentication and authorization. The evaluation of the proposed model and the implemented mechanism was conducted through functional and performance laboratory experiments, following the practice of evaluation of existing mechanisms for authentication and authorization.

Keywords: Authentication. Authorization. Access Control. Software Agent. Software Engineering. Ontologies.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

API	Application Programming Interface
ARQ	SPARQL Processor for JENA
ASF	Apache Software Foundation
BDI	Belief Desire Intention
DML	Data Manipulation Language
FIPA	Foundation for Intelligent Physical Agents
GRL	Goal-oriented Requirements Language
GWT	Google Web Toolkit
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
JADE	Java Agent Development Framework
JENA	Java framework for building Semantic Web and Linked Data applications
LDAP	Lightweight Directory Access Protocol
LOM	Learning Object Metadata
LOMS	Learning Objects Management System
MILOS	Multiagent Infraestructure for Learning Object Support
MPEG	Moving Picture Experts Group
OA	Objeto de Aprendizagem
OBAA	Objetos de Aprendizagem Baseados em Agentes Artificiais
OMT	Object-Modeling Technique
OOSE	Object-Oriented Software Engineering
OWL	Web Ontology Language
RBAC	Role-Based Access Control
RDBMS	Relational Database Management System
RDF	Resource Description Framework
RIA	Rich Internet Application
SD	Strategic Dependence
SPARQL	Protocol and RDF Query Language
SR	Strategic Reason
TDB	JENA component for RDF storage and query
UML	Unified Modeling Language
URI	Uniform Resource Identifier

W3C	World Wide Web Consortium
XML	eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

As tarefas de gerenciamento e administração são fundamentais em qualquer tipo de sistema de gerenciamento de informação. Assim, um ambiente de ensino composto por objetos de aprendizagem, usuários e aplicações que são modelados por perfis e ontologias e, integrados através de agentes de software, também requer métodos e mecanismos de gerência e administração eficazes, para lidar com este cenário complexo.

É de grande relevância no processo de gerenciamento e administração da informação a proteção dos dados e dos recursos envolvidos, de modo a garantir o acesso, alteração e liberação apenas a pessoas devidamente autorizadas. Esta questão de segurança da informação está fortemente relacionada à administração moderna representando um bem que por sua vez precisa ser protegido, buscando diminuir ao máximo os riscos no tocante ao uso indevido ou extravio da informação.

A plataforma OBAA-MILOS (VICARI; GLUZ, 2011), que tem por objetivo criar uma infraestrutura baseada em agentes e capaz de suportar o ciclo de vida completo de um objeto de aprendizagem, de acordo com a proposta de metadados OBAA (VICARI et al., 2010), é constituída por vários sistemas multi-agente capazes de auxiliarem nas atividades de autoria, busca, uso e gerência de OA.

Inserido no projeto OBAA-MILOS, este trabalho propõe um modelo de autenticação e autorização de usuários e agentes para a plataforma MILOS (GLUZ, 2010), fazendo uso de tecnologias empregadas na plataforma, como agentes inteligentes e ontologias. O resultado esperado será um modelo e uma arquitetura capaz de suportar o controle de acesso aos recursos da plataforma, levando em consideração todas as interfaces, agentes (usuários e aplicações) e recursos.

Embora os objetivos e questões envolvidas em proteção da informação sejam os mesmos tanto para sistemas tradicionais como para sistemas multi-agente, a análise, projeto e desenvolvimento difere bastante uma vez que as tecnologias e ferramentas utilizadas, muitas delas recentes e em desenvolvimento, devem compreender aspectos que se relacionam às características de um ambiente multi-agente. Isso envolve, conforme (BRESCIANI et al., 2004) a autonomia dos agentes, seus aspectos de habilidade social (comunicação), além da complexidade de reatividade e proatividade. Os aspectos relacionados à proteção e segurança da informação em um cenário multi-agente, configura um desafio a ser enfrentado durante a realização do trabalho.

Com a finalidade de validar o modelo definido e seus mecanismos de gerenciamento, foi desenvolvido um protótipo do sistema, de nome *Fayol*. O nome *Fayol*, foi inspirado em Jules Henri Fayol fundador da teoria clássica da administração (GOLEMAN, 2007) e considerado um dos principais contribuintes para o desenvolvimento do conhecimento administrativo moderno.

Este trabalho está estruturado da seguinte forma: Nas próximas seções são apresentadas a questão de pesquisa, seguida dos objetivos e da metodologia utilizada. O capítulo dois apresenta a fundamentação teórica com os conceitos e ferramentas que serão utilizadas. No capítulo três são apresentados trabalhos relacionados ao tema proposto. No capítulo quatro é descrito o mo-

delo e a arquitetura proposta para o desenvolvimento do sistema. Finalmente no capítulo cinco são apresentados os experimentos e resultados obtidos com a utilização do sistema proposto nesta dissertaão. Finalmente, o capítulo seis apresenta as conclusões do trabalho desenvolvido.

1.1 Motivação

A plataforma OBAA-MILOS (VICARI; GLUZ, 2011), composta por um conjunto de aplicações multi-agente, cada uma com seu objetivo específico, seja ele de busca, utilização, autoria ou gerência de objetos de aprendizagem, necessita de um modelo para gerenciamento de seus usuários e respectivos perfis de acesso às informações. O próprio conjunto de aplicações, deverá ser controlado com relação ao acesso a determinadas áreas da base de objetos de aprendizagem.

1.2 Questão de Pesquisa

Frameworks para controle de acesso como Spring Security e OpenLDAP não são compatíveis com a plataforma de agentes MILOS (GLUZ, 2010), tornando-se necessário a definição de um modelo de controle de acesso capaz de suportar as principais tecnologias da MILOS.

A questão de pesquisa do trabalho resume-se a como agentes de software, utilizando uma base semântica, inseridos na plataforma MILOS (GLUZ, 2010) e portanto, fazendo uso de tecnologias já existentes no ambiente, poderiam compor um modelo de autenticação e autorização de usuários e agentes para esta plataforma. Este modelo deve operacionalizar mecanismos semânticos de controle de acesso que possam fornecer a segurança e a confiabilidade necessários de forma eficaz e aderente às características da plataforma MILOS (VICARI; GLUZ, 2011).O resultado esperado será um modelo arquitetural e um protótipo de sistema de controle de acesso e autentica cão capaz de suportar a gerência e o controle de acesso aos recursos da plataforma, levando em consideração todas as interfaces, agentes (usuários e aplicações) e recursos existentes, utilizando a mesma base ontológica já utilizada nesta plataforma.

1.3 Objetivos

Dentre as possíveis soluções pesquisadas, como veremos nos capítulos e seções posteriores, chegou-se à conclusão que a melhor alternativa seria o desenvolvimento de um modelo próprio para realizar a gerência e o controle de acesso da plataforma MILOS. Desta forma, o principal objetivo computacional do trabalho será a criação de um sistema de gerência e controle de acesso, utilizando as tecnologias de agentes inteligentes de software e uma base ontológica para descrever os usuários (agentes e aplicações), os perfis, os recursos e as formas de acesso a estes recursos.

Este objetivo geral pode ser dividido nas seguintes etapas e seus objetivos específicos.

a) Criar um modelo ontológico para os elementos que compõe o sistema de controle de

acesso da plataforma MILOS, incluindo usuários, perfis de usuários, aplicações, perfis de aplicações e recursos como objetos de aprendizagem.

b) Criar um modelo dinâmico baseado em agentes de software que implemente a funcionalidade de gerência de usuários e controle de acesso federado, com ou sem sessão para o domínio da plataforma MILOS.

c) Verificar a eficácia do modelo através de experimentos controlados de laboratório.

d) Verificar a aderência do modelo à plataforma MILOS, através de experimentos integrados ao seu ambiente de agentes, aplicações e objetos de aprendizagem.

Estes objetivos estão alinhados com objetivos importantes do projeto OBAA-MILOS, enfatizando a necessidade de acesso federado e sem necessidade de sessão, em razão da distribuição em vários ambientes de execução previstos para a plataforma MILOS. O modelo e os agentes desenvolvidos nessa dissertação serão partes importantes da infraestrutura de agentes resultante desse projeto, sendo responsável pelos serviços de autenticação e autorização da plataforma.

1.4 Metodologia

A metodologia utilizada neste projeto seguiu as diretrizes técnicas e metodológicas adotadas na infra-estrutura MILOS (GLUZ, 2010) que prevê o uso de camadas de ontologias, sistemas de agentes inteligentes e facilidades de interface.

O protótipo desenvolvido foi utilizados para a realização de todos os testes de laboratório, incluindo a manutenção de usuários e perfis, além da simulação de cenários de acessos. O ambiente da plataforma MILOS, seus agentes, aplicações e recursos da base semântica foram utilizados para receber a ontologia de acesso criada e a configuração para a realização dos experimentos de avaliação.

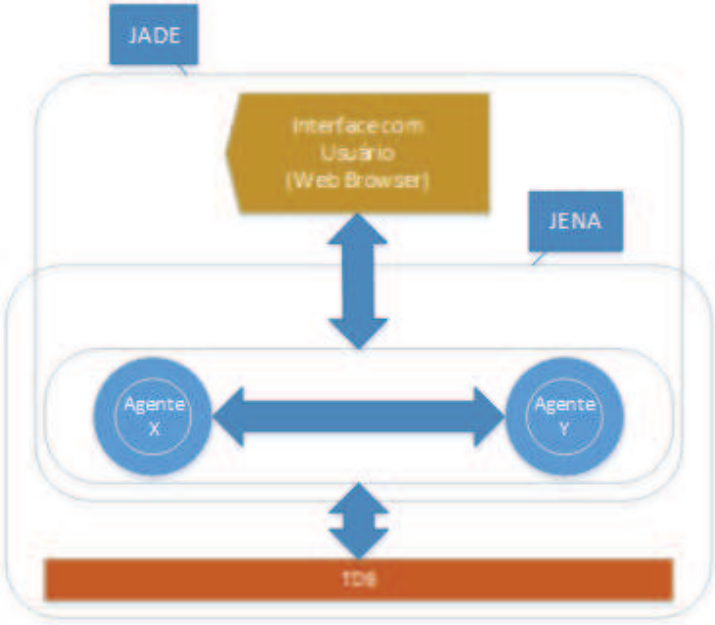
O trabalho utilizou, no decorrer do desenvolvimento, diversas tecnologias e ferramentas já utilizadas na plataforma. Entre elas destacamos:

- A metodologia Tropos combinada com diagramas de UML para as especificações não funcionais e funcionais respectivamente.
- Agentes de software, criados em linguagem Java com a utilização dos *frameworks* JADE e JENA com o objetivo de facilitar a comunicação entre eles e a manipulação e persistência de ontologias no formato RDF.
- Software Protégé para a criação e validação das ontologias de usuários, perfis, aplicações e acesso.
- Uma interface gráfica para a realização das tarefas de gerenciamento e administração de usuários e recursos, utilizando os agentes de acesso à base semântica.

- Uma ontologia de acesso com o objetivo de definir o modelo de dados, seus conceitos e os relacionamentos entre eles.

A Figura 1 mostra o relacionamento dos *frameworks* JADE e JENA na estrutura do sistema *Fayol*.

Figura 1 – Visualização da utilização dos frameworks JADE e JENA



Fonte: Elaborada pelo autor

A linguagem JAVA e os *frameworks* JADE e JENA foram escolhidos por sua versatilidade, eficiência, portabilidade, segurança e também por já fazerem parte da plataforma MILOS existente.

Com o objetivo de garantir a segurança na troca de mensagens entre os agentes do sistema foi utilizado JADE-S, um *add-on* da plataforma JADE para esta finalidade.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta as bases tecnológicas sobre as quais o sistema *Fayol* está fundamentado.

O sistema *Fayol* se contextualiza como um sistema de controle de acesso para a plataforma MILOS (GLUZ, 2010), que pretende fornecer o suporte completo ao ciclo de vida dos objetos de aprendizagem OBAA (VICARI et al., 2010) e as tecnologias de agentes inteligentes, mais especificamente à infraestrutura de agentes MILOS (VICARI; GLUZ, 2011), construída para aceitar as características e funcionalidades dos objetos de aprendizagem compatíveis com o padrão OBAA.

Assim, a área mais geral onde o *Fayol* se insere é a área de Segurança da Informação, implementando o protótipo necessário para validar o modelo de controle de acesso apresentado nesta dissertação. Para esta implementação, serão utilizadas tecnologias de processamento de ontologias, incluindo mecanismos de representação e manipulação de metadados em uma base de dados ontológica. Tais bases farão uso de triplas RDF, que é o formato *nativo* das ontologias OWL, como formato de representação e manipulação de informações. Tais tecnologias específicas também são descritas nas próximas seções.

Algumas tecnologias utilizadas estão consolidadas atualmente e são adotadas com sucesso em diversos projetos, com documentação e estudo de casos detalhados. Outras tecnologias porém, são emergentes e, em alguns casos, ainda em desenvolvimento. O uso destas tecnologias criou uma demanda de análises mais aprofundadas dos resultados, principalmente com relação ao comportamento que o sistema apresentou quando inserido no ambiente de agentes já existente.

2.1 Segurança de Sistemas

Pensar em segurança de um sistema de uma forma abstrata exige uma definição do seu comportamento com relação a agentes que atuam sobre objetos, o que em última análise representa a operação do sistema. A questão relacionada à segurança de um sistema procura determinar e definir que agentes poderão alterar o estado de um sistema durante o seu funcionamento, especificando a cada agente um ou mais privilégios sobre um ou mais recursos do sistema.

Fica claro então que para a criação de qualquer modelo de segurança em um sistema precisamos identificar (autenticar) agentes, recursos e privilégios ou modos de acesso que serão utilizados para sua operação.

2.1.1 Autenticação

O processo de autenticação é, na maioria dos casos, o primeiro passo para o uso de qualquer sistema, principalmente em ambientes de rede onde existe compartilhamento de recursos

e informações.

Dentre os esquemas de autenticação, destacam-se usuário-senha, utilização de *tokens* e mais recentemente, esquemas biométricos (RAMACHANDRAN, 2002).

Os esquemas usuário-senha, conhecidos como esquemas de fator único, são de longe os mais utilizados. O sistema atribui um identificador (usuário) para cada agente e vale-se de uma informação que o agente conhece (senha) para realizar a autenticação.

Os esquemas que utilizam *tokens* para autenticação são também conhecidos como esquemas de duplo fator, pois autenticam um usuário com base em algo que o usuário conhece (senha) e em algo que o usuário possui (token). Os dois esquemas mais populares baseados em *tokens* são SecurID e cartões inteligentes.

a) SecurID utiliza um algoritmo proprietário que executa no *token* do usuário e em um servidor corporativo de autenticação de *token*.

b) Cartões inteligentes são dispositivos com capacidade de processamento, no formato e tamanho de cartões de crédito. Geralmente possuem co-processadores criptográficos para suportar criptografia forte que não seria possível utilizando-se processadores comuns. Os cartões inteligentes suportam autenticação unidirecional simétrica, autenticação mútua simétrica e autenticação estática ou dinâmica assimétrica.

Os esquemas de autenticação biométricos são também chamados de esquemas de autenticação de três fatores e realizam a autenticação de usuários com base em algo que eles sabem (senha), algo que eles possuem (token) e algo que eles são (escaneamento de retina, impressão digital, ou varredura térmica). Estes esquemas são muito fortes porém caros e, portanto, inaplicável em muitos cenários. Existem esquemas diferenciados de autenticação biométrica, onde a autenticação pode ser realizada apenas com o terceiro fator de autenticação (retina ou digital) ou uma combinação com o primeiro ou segundo fatores.

Mais recentemente, alguns esquemas utilizam-se de um quarto fator (ABHISHEK et al., 2012), relacionado a "alguém que o usuário conheça": Fator Social, para a autenticação do usuário.

2.1.2 Autorização

Muitas vezes referenciado como controle de acesso, é responsável por garantir que todas as solicitações serão tratadas de acordo com a política de segurança estabelecida para o sistema ao qual se aplica. A política de segurança define regras e modos de operação permitidos dentro do sistema. Normalmente, qualquer ação que não esteja explicitamente definida por uma política de segurança, será negada.

O núcleo de um modelo de controle de acesso é a capacidade de tomar decisões de acesso. Uma decisão de acesso deve ser tomada sempre que um agente (usuário ou aplicação) solicita acesso a um objeto (recurso). Toda a decisão requer dois componentes: Os atributos de dados

descrevendo o agente e o objeto alvo da ação. O processo de decisão combina esta informação para se chegar a um "Permitido"ou "Negado"como resposta.

Alguns modelos de controle de acesso permitem a negociação da solicitação, solicitando informações adicionais ao agente antes de tomar a decisão final de resposta.

2.1.3 Modelos de Controle de Acesso ou Autorização

De acordo com (RAMACHANDRAN, 2002), um modelo de controle de acesso deve fornecer uma referência de alto nível, independente de domínio e de implementação para a arquitetura e projeto de mecanismos de acesso. Historicamente, os modelos de controle de acesso são classificados em duas grandes categorias: obrigatórios e discricionários.

2.1.3.1 Modelo Obrigatório

Controle de acesso obrigatório orienta o acesso a objetos usando uma classificação hierárquica de rótulos. A cada agente e objeto é atribuído um rótulo. Todo o acesso é controlado com base em comparações destes rótulos e, em geral, é aplicada estaticamente. Dizemos que o controle de acesso é obrigatório porque ele é centralizado no sistema, todas as decisões relacionadas a atividades de agentes com base em rótulos individuais. A propriedade de objetos não constitui fator de importância para atribuição ou delegação de acesso. O controle de acesso obrigatório centraliza a base de conhecimento usada para tomar decisões, embora agentes e objetos possam negociar o acesso com base em informações locais.

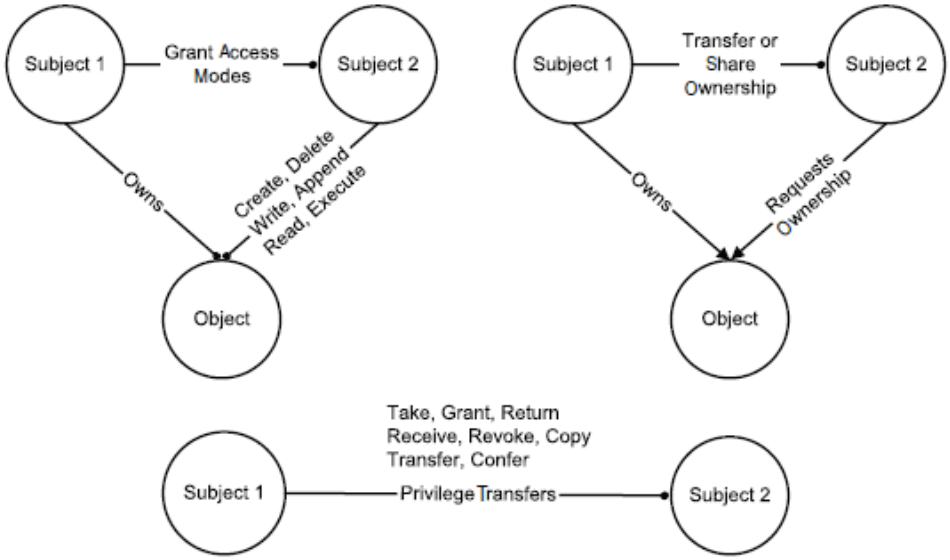
2.1.3.2 Modelo Discricionário

Controle de acesso discricionário é aquele baseado na matriz de acesso de Lampson (LAMPSON, 1974), que organiza a segurança de um sistema em uma matriz bidimensional de autorizações onde cada par agente-objeto corresponde a um conjunto de modos de acesso permitidos.

Controle de acesso discricionário orienta o acesso a objetos baseado na propriedade do objeto ou credencial de delegação fornecida pelo agente. Estes modelos são implicitamente dinâmicos na medida em que permitem que os usuários concedam e revoguem privilégios para outros usuários ou entidades. A Figura 2 apresenta estes relacionamentos de concessão de privilégios.

O modelo discricionário é um modelo simplificado, restrito apenas a agente e objeto, não existe neste modelo o conceito de papéis. Trata-se de um modelo flexível, porém a propagação de direitos através do sistema pode ser complexa para fins de rastreamento e pode criar paradoxos do tipo: Se o agente A concede a um agente B o acesso "compartilha propriedade"a um objeto, e B, por sua vez concede a C o acesso "ler", o que acontece quando A revoga o privilégio concedido a B? Neste caso o sistema deverá decidir o que fazer entre revogar acessos em

Figura 2 – Concessão de permissão de acesso e propriedade



Fonte: (RAMACHANDRAN, 2002)

cascata ou rejeitar a revogação até que B revogue o que delegou a C.

2.1.3.3 Modelo Baseado em Papéis

O modelo baseado em papéis ou funções (RBAC - Role-based Access Control), é o modelo dominante tanto na pesquisa acadêmica como em produtos comerciais. Teve sua aceitação generalizada devido à sua arquitetura simplificar a administração de segurança, incluindo herança semântica de papéis e permitindo uma definição de política de segurança abrangente, com fácil revisão das atribuições agente-papel e papel-permissão. A Figura 3 apresenta o modelo RBAC plano, segundo (FERRAIOLO; KUHN; CHANDRAMOULI, 2003).

Figura 3 – Modelo RBAC Plano

Fonte: (FERRAIOLO; KUHN; CHANDRAMOULI, 2003)

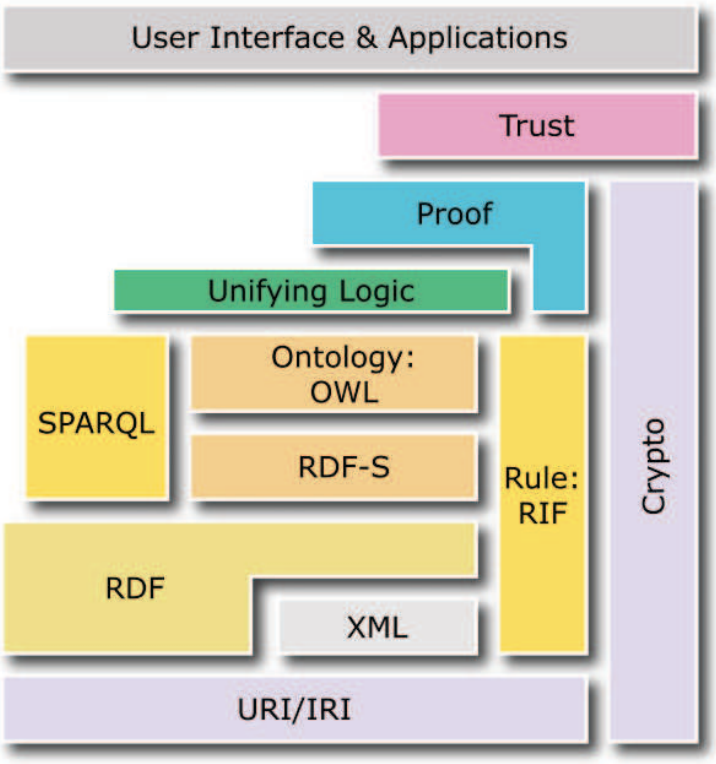
Este trabalho cria um modelo RBAC com uma das sugestões apresentas em (FERRAIOLO; KUHN; SANDHU, 2007), que é a autorização sem o uso de sessões, utilizando agentes inteligentes de software e uma ontologia de acesso para realizar os serviços de autenticação e autorização para a plataforma MILOS.

2.2 Web Semântica

A Web Semântica é um movimento colaborativo liderado pelo organismo de normalização internacional W3C [1], que define padrões para formatos de dados, incentivando a inclusão de conteúdo semântico em páginas da Web.

O termo *Web Semântica* foi criado por (SHADBOLT; BERNERS-LEE; HALL, 2006) para definir uma Web de dados interligados que poderiam ser processados por máquinas. A Figura 4 apresenta a arquitetura da Web Semântica segundo Tim Berners-Lee.

Figura 4 – Arquitetura da Web Semântica



Fonte: http://www.w3.org/2007/Talks/0130-sb-W3CTechSemWeb

A principal meta da Web Semântica é converter a web atual, dominada por documentos não estruturados e semi-estruturados em uma *rede de dados*, o que permitiria o compartilhamento e a reutilização de forma automatizada por aplicações (sistemas), empresas e comunidades.

2.2.1 Ontologia

De acordo com (GÓMEZ-PÉREZ; FERNANDEZ-LOPEZ; CORCHO, 2004) existem diversas definições de ontologia, muitas vezes associadas e dependentes do processo que seu autor utilizou para sua criação. Resumidamente consideram que uma ontologia visa capturar o conhecimento consensual de uma forma genérica, e que pode ser reutilizada e compartilhada

[1] http://www.w3.org/

entre aplicações de software e por grupos de pessoas.

Ontologias geralmente são criadas por mais de uma pessoa com amplo conhecimento sobre o objeto alvo de sua criação.

A definição de ontologia utilizada neste trabalho será a de um modelo de dados que representa um conjunto de conceitos dentro de um domínio e os relacionamentos entre estes conceitos.

2.2.2 Ferramentas

A seguir descrevemos algumas ferramentas que foram utilizadas para tratar, representar e manipular ontologias.

2.2.2.1 JENA

JENA (JENA, 2013) é um *framework* de código fonte aberto e livre, voltado para a construção de aplicações de web semântica. JENA tem suporte da ASF e compreende APIs escritas em Java, para manipulação, armazenamento e consultas de RDFs e tratamento de ontologias (OWL - Modelagem e inferências). A Tabela 1 apresenta a divisão das APIs JENA, de acordo com a sua atuação.

Tabela 1 – Framework JENA

RDF	**Armazenamento de Triplas**	**OWL**
RDF API	TDB	API de Ontologia
ARQ (SPARQL)	Fuseki	API de Inferência

Fonte: (JENA, 2013)

A Figura 5 apresenta a interação entre as diversas APIs do *framework* JENA.

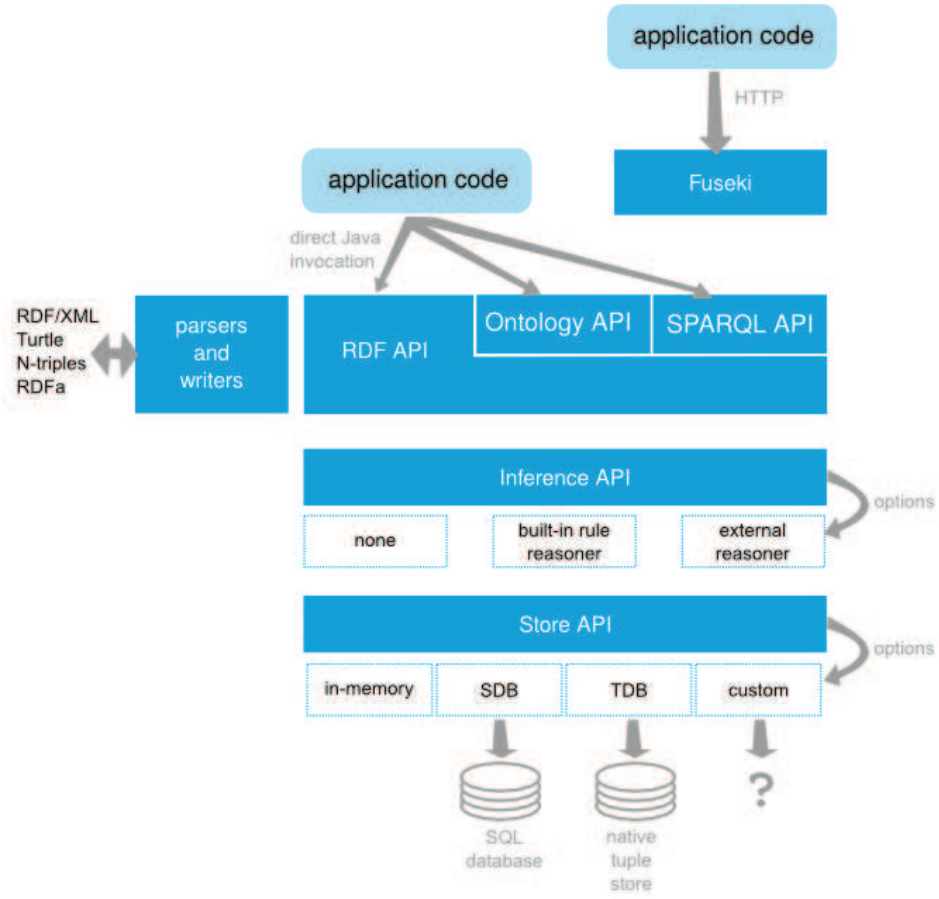
2.2.2.2 TDB

TDB é um componente do *framework* JENA destinado ao armazenamento e consulta de triplas RDF. Serve de apoio aos demais componentes, sendo utilizado para persistência RDF alto desempenho numa única máquina.

Uma base de dados TDB pode ser acessada e gerenciada com a utilização de scripts de linha de comando ou através da API incluída no *framework* JENA.

A Figura 6 apresenta um exemplo de consulta a uma base de dados TDB utilizando localização de arquivos por diretório.

Figura 5 – Interação Entre APIs JENA



Fonte: http://jena.apache.org

Figura 6 – Exemplo de uso da API TDB

```
public static void exemplo2() {
    // Direct way: Make a TDB-back Jena model in the named directory.
    String directory = "C:\\tdbteste\\tdb1" ;
    Dataset ds = TDBFactory.createDataset(directory) ;
    Model model = ds.getDefaultModel() ;

    com.hp.hpl.jena.query.Query q = QueryFactory.create("");

    // Create a SPARQL-DL query execution for the given query and
    // ontology model
    QueryExecution qe = SparqlDLExecutionFactory.create(q, model);

    // We want to execute a SELECT query, do it, and return the result set
    ResultSet rs = qe.execSelect();

    System.out.print("Retornou -> " + rs.getResourceModel().getGraph());

    // Close the dataset.
    ds.close();
}
```

Fonte: Elaborada pelo autor

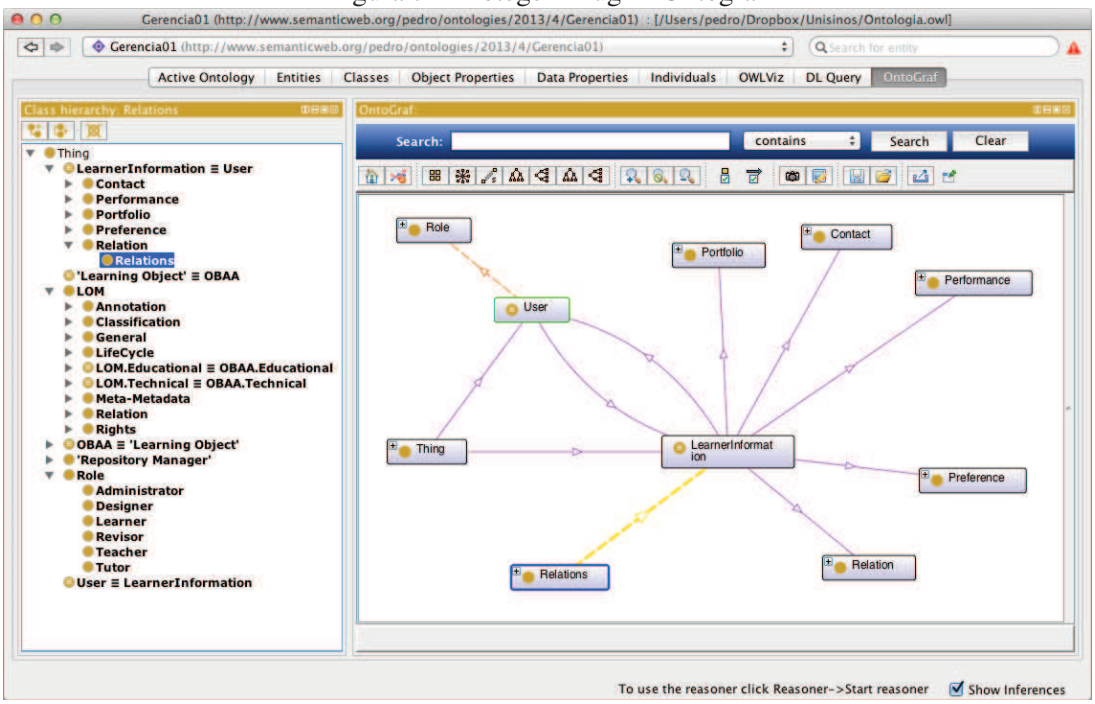
2.2.2.3 Protégé

O Protégé é um software escrito em Java para criação e modelagem de ontologias. Possui um interface gráfica que facilita a criação, visualização, edição e exportação de modelos de ontologias em formato RDF, OWL e XML [2].

Algumas características do Protégé, como possibilidade de inclusão de *plug-ins* escritos em Java, execução de *reasoners*, edição de indivíduos para marcação de Web Semântica entre outras, tornam o Protégé uma ferramenta amplamente utilizada por desenvolvedores e pesquisadores em diversas áreas do conhecimento.

A Figura 7 apresenta a tela de visualização do *plug-in Ontograf* do Protégé na ontologia de usuário.

Figura 7 – Protégé - Plug-in Ontograf



Fonte: Elaborada pelo autor

2.2.3 Linguagens

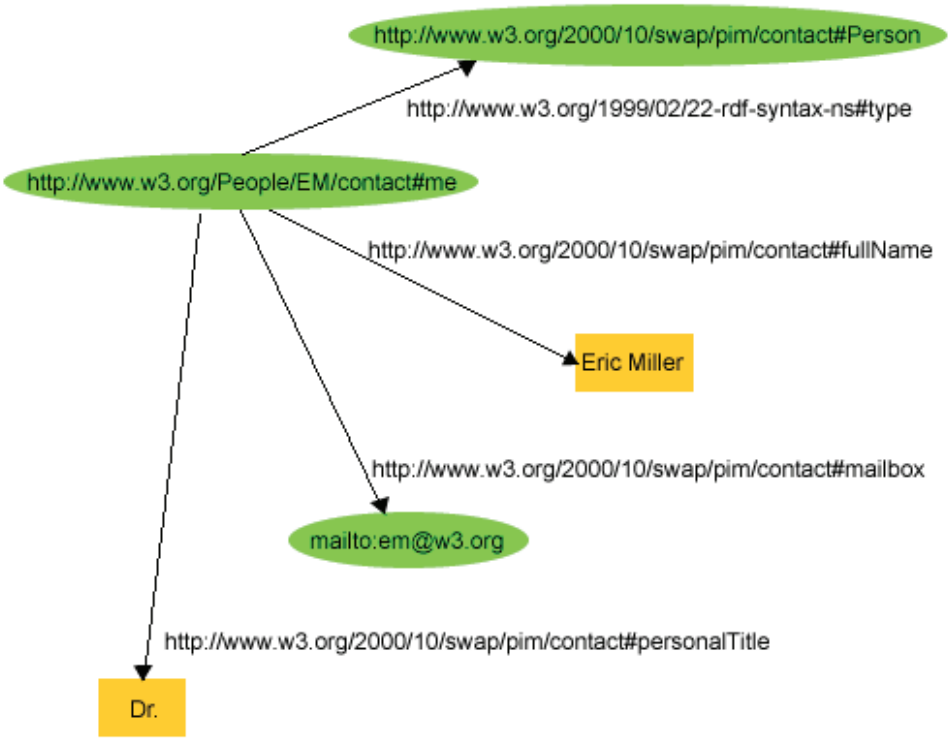
A seguir descrevemos as linguagens utilizadas para manipulação de dados e metadados gerenciados pelo sistema *Fayol*.

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

2.2.3.1 RDF

RDF (Resource Description Framework) (GANDON; SCHREIBER, 2014) é um conjunto de especificações do W3C, criado com o objetivo de modelar dados de metadados. Tem sido utilizado como um método de descrição conceitual ou modelagem de informação, utilizando uma variedade de notações de sintaxe e formatos de serialização de dados.

Figura 8 – Grafo RDF



Fonte: http://www.w3.org

Assemelha-se às abordagens de modelagem conceitual clássicas como entidade-relacionamento ou diagramas de classe, uma vez que se baseia na ideia de fazer declarações sobre os recursos (em especial os recursos da web) na forma de expressões sujeito - predicado - objeto. Essas expressões são conhecidas como tripas RDF.

RDF é ainda, um modelo padrão para o intercâmbio de dados na web por possuir características que facilitam a fusão de dados, mesmo que os esquemas subjacentes sejam diferentes, permitindo a evolução dos esquemas ao longo do tempo sem a necessidade de alteração de seus consumidores.

RDF estende a estrutura de ligação da Web utilizando URIs para nomear a relação entre os objetos e as duas extremidades da ligação (isto é, normalmente referido como uma "tripla"). A utilização deste modelo permite que os dados estruturados e semi-estruturados sejam misturados, expostos e compartilhados entre diferentes aplicações.

A estrutura de ligação do RDF forma um grafo, onde as arestas representam o elo de ligação

entre dois recursos, representados pelos nós de gráfico. Um exemplo pode ser visto na Figura 8 onde se pode verificar que "há uma pessoa identificada por http://www.w3.org/People/EM/contact#me, cujo nome é Eric Miller, cujo endereço de email é em@w3.org, e cujo título é Dr.".

2.2.3.2 OWL

Utilizada para definir e instanciar ontologias, OWL (MOTIK; PARSIA; PATEL-SCHNEIDER, 2012) pode ainda incluir descrições de classes, suas respectivas propriedades e seus relacionamentos. Por possuir um vocabulário adicional com uma semântica formal, a utilização de OWL permite às aplicações o processamento, a interpretação e a troca de informações entre elas.

OWL foi projetada para uso por aplicações que precisam processar o conteúdo da informação de forma automatizada em vez de apenas apresentar estas informações. A utilização de OWL facilita a interpretação do conteúdo da informação por um sistema computacional se comparado a conteúdos que utilizem apenas o formato XML ou RDF por exemplo, fornecendo vocabulário adicional juntamente com uma semântica formal.

OWL possui três variantes ou sub-linguagens que são OWL Lite, OWL DL e OWL Full.

- OWL Lite fornece suporte a necessidades de classificação hierárquica e restrições simples. Embora suporte restrições de cardinalidade, só permite valores de cardinalidade 0 ou 1. OWL Lite também tem uma menor complexidade formal que OWL DL.

- OWL DL (DL - Description Logic) Suporta máxima expressividade, enquanto mantém-se computável, isto é, garante que todas as conclusões de sua descrição sejam computáveis e conclusivas em um tempo finito. OWL DL inclui todas as construções da linguagem OWL, porém elas somente podem ser usadas com algumas restrições (por exemplo, embora uma classe possa ser subclasse de muitas classes, uma classe não pode ser instância de outra classe).

- OWL Full garante a máxima expressividade e liberdade sintática do RDF sem nenhuma garantia computacional. Em OWL Full uma classe pode ser tratada simultaneamente como uma coleção de indivíduos e como um indivíduo em si mesmo. OWL Full permite que uma ontologia aumente o vocabulário pré-definido de RDF ou OWL.

O *framework* JENA dispõe uma API destinada à manipulação de ontologias escritas em OWL. Esta API, em sua última versão, fornece suporte a escrita e leitura de ontologias nos formatos RDF/XML, OWL/XML e Turtle, além de ser possível a integração com motores de inferência disponíveis, como: Pellet, FaCT++, HermiT e RacerPro.

2.2.3.3 SPARQL Query e Update

SPARQL Query (SEABORNE; HARRIS, 2013) e SPARQL Update (POLLERES; GEARON; PASSANT, 2013) formam um conjunto de especificações e protocolos que fornecem

uma DML (Data Manipulation Language) para manipulação de dados armazenados no formato RDF. Para a utilização de SPARQL em um conjunto de dados RDF, é necessário o conhecimento prévio da ontologia que os dados representam. A Figura 9 apresenta um exemplo de consulta SPARQL.

Figura 9 – Exemplo Consulta SPARQL

Fonte: http://www.w3.org

2.3 Sistemas Multi-Agentes

Segundo (WOOLDRIDGE, 2008), um sistema multi-agente é formado por dois ou mais agentes, interagindo entre si através de comunicação. Os agentes podem ainda atuar em ambientes diferentes, o que o autor chama de "esferas de influência", no sentido de que eles terão o controle ou pelo menos a capacidade de influenciar diferentes partes do ambiente. Em alguns casos, estas "esferas de influência"podem coincidir, gerando relações de dependência. Um sistema multi-agente pode ainda configurar relacionamentos de hierarquia entre os agentes, onde um agente pode assumir o papel de chefe e outro de subordinado.

Em um ambiente multi-agente, algumas características como dados e controle distribuídos, diversidade de conhecimento, objetivo global decomposto em objetivos a ser atingidos por cada agente, multiplicidade de funções e um certo grau de autonomia devem ser encontrados.

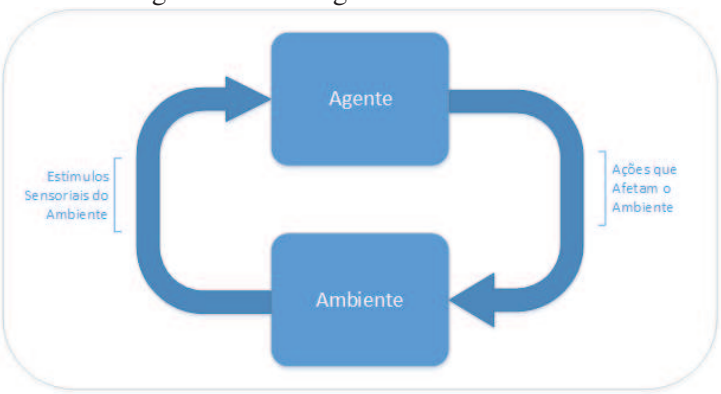
O presente trabalho propõe a utilização de uma metodologia mista, descrita posteriormente, composta por elementos de UML e Tropos para a especificação de requisitos e desenvolvimento do sistema *Fayol*.

2.3.1 Agentes de Software

De acordo com (WOOLDRIDGE, 2008), um agente é um sistema computacional capaz de realizar ações autonomamente para atender seus objetivos de projeto, ao invés de simplesmente obedecer às regras sobre o que fazer em qualquer momento. A representação de um agente e seu ambiente pode ser observada na Figura 10.

Um agente então, é um módulo de software independente, que percebe eventos do ambiente onde está inserido, recebe mensagens ou solicitações, executa ações e eventualmente envia mensagens de caráter informativo ou mesmo solicitações a outros módulos ou agentes do ambiente, buscando atingir um objetivo específico.

Figura 10 – Um agente no seu ambiente



Fonte: (WOOLDRIDGE, 2008)

2.3.2 Ferramentas

2.3.2.1 JADE

JADE (Java Agent Development Framework) é um *framework* implementado na linguagem Java que tem por objetivo simplificar a implementação de sistemas multi-agente. JADE está em conformidade com as especificações da FIPA fornecendo um conjunto de ferramentas gráficas que suportam as fases de implantação e depuração de agentes JAVA.

JADE permite a utilização de seus componentes diretamente na camada de interface, possibilitando assim a interação das ações de usuários com o ambiente de agentes.

2.3.2.2 Infraestrutura MILOS

A infraestrutura MILOS fornece suporte aos processos de autoria, gerência, busca e disponibilização de objetos de aprendizagem de acordo com a proposta de padrão de metadados OBAA (VICARI et al., 2010).

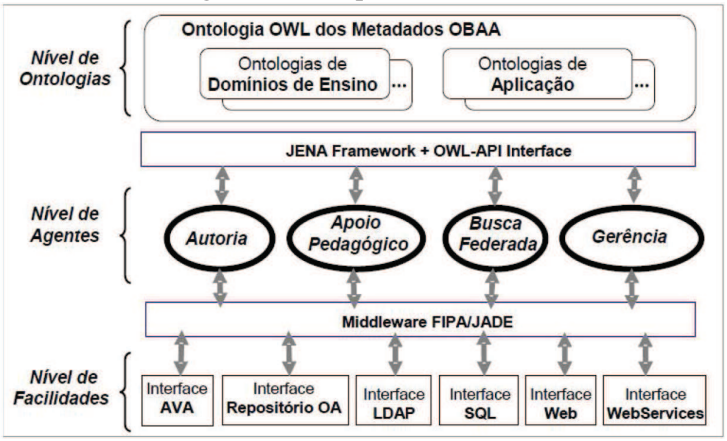
Composta de três camadas de abstração, conforme a Figura 11, o objetivo a longo prazo da MILOS é especificar e implementar uma arquitetura de agentes que seja capaz de suportar os requisitos de adaptabilidade, interoperabilidade e acessibilidade previstos pela proposta OBAA.

O nível de ontologias tem como papel principal representar os conhecimentos necessários a serem compartilhados pelos agentes, especificando as propriedades e ações do domínio da aplicação.

O nível de agentes desempenha o papel ativo na arquitetura. É nesta camada que são implementadas todas as funcionalidades e serviços disponibilizados pela infraestrutura. Esses agentes fazem uso das ontologias do primeiro nível, para dar o suporte necessário e atender os requisitos dos subsistemas existentes na plataforma.

O nível de facilidades define os serviços que dão suporte à interoperabilidade da infraestru-

Figura 11 – Arquitetura MILOS



Fonte: Gluz e Vicari (2010)

tura, através de uma plataforma de comunicação entre agentes. Esta camada oferece facilidades de interface permitindo que os agentes e suas aplicações possam ser acessados por diferentes plataformas operacionais da Web.

2.3.3 Engenharia de Software Orientada a Agentes

Segundo (CASTRO; ALENCAR; SILVA, 2006) um dos mais promissores paradigmas para aumentar a habilidade dos engenheiros de software em melhorar o entendimento das características de um software complexo, onde se reconhece a interação como um dos aspectos mais importantes, é a orientação a agentes.

"É a naturalidade e a facilidade com que uma variedade de aplicações pode ser caracterizada em termos de agentes que levam pesquisadores e desenvolvedores a ficarem tão entusiasmados sobre o potencial deste paradigma. (CASTRO; ALENCAR; SILVA, 2006)".

2.3.3.1 UML

Originada como uma seleção de boas práticas de engenharia de software, derivada da fusão de conceitos de OMT e OOSE (BOOCH; JACOBSON; RUMBAUGH, 2000), a UML é uma linguagem de modelagem comum e largamente utilizada que pretende tornar-se um padrão para modelar sistemas concorrentes e distribuídos.

Apesar de atualmente, a versão 2.2 da UML apresentar quatorze tipos de diagramas estruturais e comportamentais, pretende-se utilizar, para a especificação funcional do sistema *Fayol*, *Use Cases*, *Diagramas de Comunicação* e *Diagramas de Sequência*.

2.3.3.2 i-Star (i*)

O nome i* (pronuncia-se eye-star) refere-se ao conceito de intencionalidade distribuída. Trata-se de um *framework* que propõe uma abordagem orientada a agentes para engenharia de requisitos centrada nas características intencionais do agente.

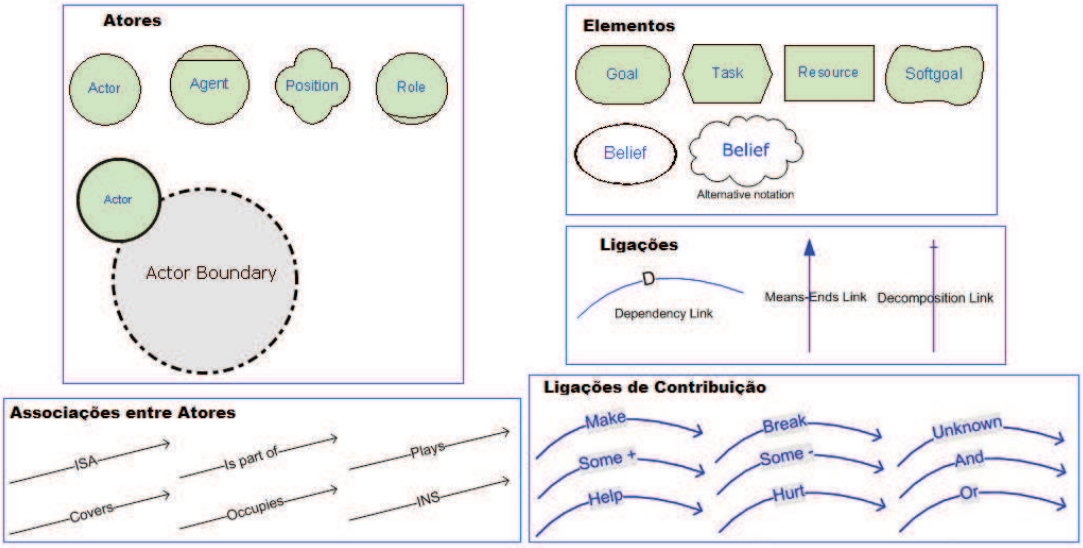
Para (YU, 2009), agentes possuem propriedades intencionais (como objetivos, crenças, habilidades, compromissos) relacionadas a outros agentes e raciocinam sobre estes relacionamentos estratégicos.

Dependências entre agentes dão origem a oportunidades, bem como vulnerabilidades. Redes de dependências são analisadas através de uma abordagem de raciocínio qualitativo. Agentes consideram configurações alternativas de dependências para avaliar o seu posicionamento estratégico em um contexto social.

i-Star é utilizado em contextos onde há várias partes (ou unidades autônomas - agentes) com interesses estratégicos que podem ser conflitantes ou que reforçam a relação entre as partes.

Elementos de i-Star, podem ser vistos na Figura 12.

Figura 12 – Elementos de i-Star



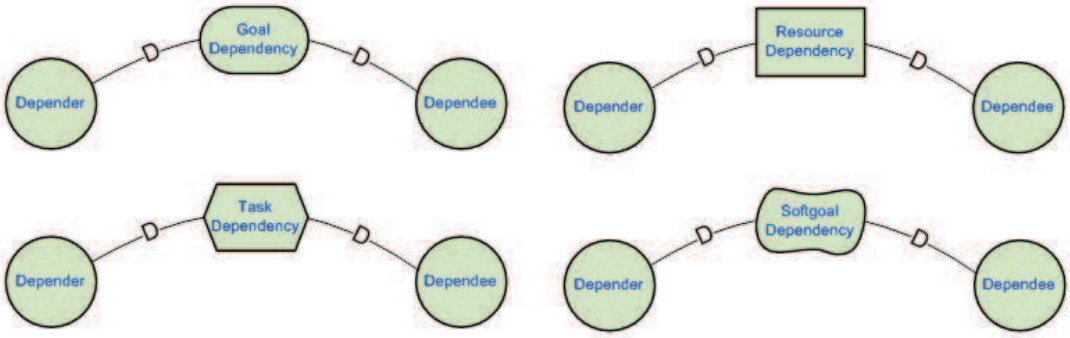
Fonte: (YU, 2009)

As dependências estratégicas representadas pelo *framework* i-Star podem ser observados na Figura 13.

2.3.3.3 Tropos

De acordo com (CERVENKA, 2012), Tropos é uma metodologia de desenvolvimento de software orientada a agentes que se baseia nos conceitos de requisitos orientados a objetivos e adotados a partir do i-Star e GRL (ITU-T, 2012). Logo, Tropos toma como ponto de partida o *framework* e os conceitos de intencionalidade distribuída do i-Star.

Figura 13 – Dependências Estratégicas i-Star



Fonte: (YU, 2009)

A modelagem Tropos trata especificamente dos requisitos e aspectos intencionais dos agentes. Esta abordagem é aplicada a arquiteturas de agentes BDI. Inspirada em conceitos organizacionais, Tropos procura reduzir o impacto das diferenças existentes entre o sistema e seu ambiente. Na fase inicial, os conceitos organizacionais são usados para modelar os requisitos iniciais, complementando com propostas para plataformas de programação orientadas a agentes. O processo começa com um modelo do ambiente no qual o sistema vai operar, utilizando na sua descrição os conceitos de atores, suas metas e interdependências. Através de refinamentos incrementais, este modelo é estendido para incluir tanto o sistema a ser desenvolvido quanto os seus subsistemas, que também são representados como atores a quem foram delegadas metas para atingir, planos para executar e recursos para fornecer.

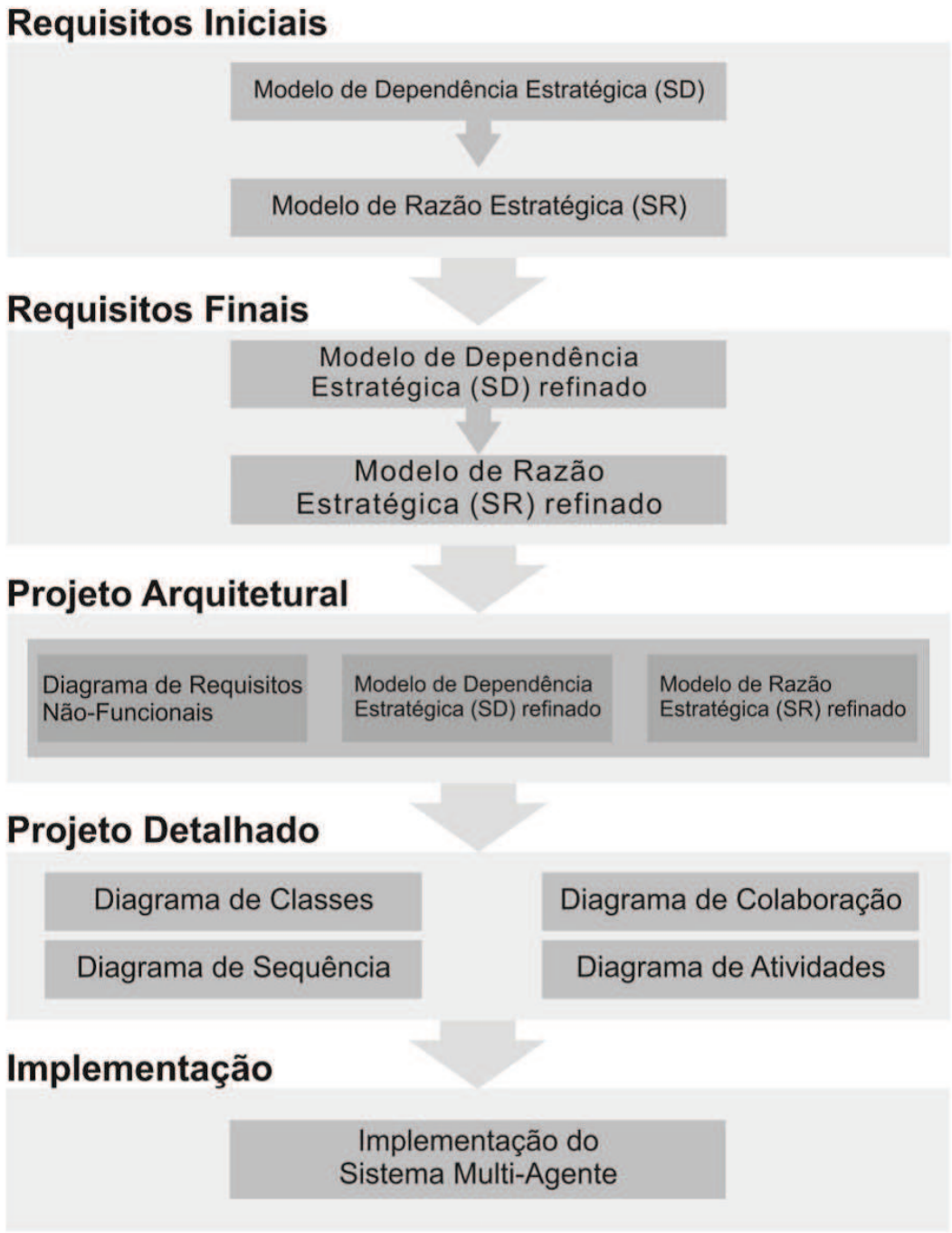
De acordo com (BRESCIANI et al., 2004), a análise e especificação de requisitos em Tropos divide-se em cinco fases ou etapas apresentadas na Figura 14:

- Requisitos Iniciais, onde se procura entender uma configuração organizacional existente. Durante esta fase, são modelados os *stakeholders* como atores e suas intenções como metas. Cada meta é analisada do ponto de vista do seu ator resultando em um conjunto de dependências entre pares de atores. As saídas desta fase são dois modelos, de dependência estratégica e de razão estratégica.

- Requisitos Finais, onde o sistema a ser desenvolvido passa a ser tratado como um outro ator no modelo de dependência estratégica, relacionando-o aos demais atores. Este novo modelo é analisado e refinado, resultando os modelos de dependência estratégica e de razão estratégica refinados.

- Projeto arquitetural e detalhado focam na especificação do sistema, de acordo com os requisitos obtidos nas fases anteriores. O projeto arquitetural define a arquitetura global do sistema com seus sub-sistemas interconectados através de dados e fluxos de controle. Sub-sistemas são representados como atores e as interconexões de dados e fluxo de controle são representadas como dependências. O projeto arquitetural prove também um

Figura 14 – Fases da Metodologia Tropos e suas Saídas



Fonte: (CASTRO; ALENCAR; SILVA, 2006)

mapeamento dos atores do sistema com um ou mais agentes de software, cada um deles caracterizado por suas funções específicas.

- Projeto detalhado auxilia na especificação das funções e capacidades dos agentes e suas interações. Neste ponto a plataforma de implementação já foi selecionada, devendo ser considerada no detalhamento do projeto com relação à codificação.

- Atividade de implementação é o passo natural que consiste no mapeamento do modelo de agentes definidos e seus respectivos códigos.

Modelos Tropos abrangem apenas os requisitos não-funcionais do ciclo de desenvolvimento de sistemas multi-agente e, parcialmente também, alguns aspectos organizacionais. Para ser capaz de cobrir outros aspectos da modelagem de sistemas multi-agente, Tropos explora diagramas de classe, atividade e diagramas de interação da UML.

3 TRABALHOS RELACIONADOS

Neste capítulo apresentaremos alguns trabalhos relacionados ao tema. O alvo inicial da pesquisa destes trabalhos concentrou-se em controle de acesso realizado por sistemas multi-agentes. Também buscou-se identificar sistemas que utilizassem ontologias para representar as informações de controle. Finalmente, procuramos por ferramentas e projetos consolidados que ofecerem parcial ou totalmente os serviços de controle de acesso. Destacamos dois trabalhos apresentados em forma de artigos, um *framework* inserido em uma plataforma para desenvolvimento de aplicações Java e um conjunto de classes para utilização de uma versão aberta do LDAP. Estes trabalhos são apresentados a seguir.

3.1 Onto-ACM (Ontology based Access Control Model) for Security Policy Reasoning in Cloud Computing

Dentre os trabalhos encontrados, (CHOI; CHOI; KIM, 2014) apresenta um modelo de análise semântica que propõe resolver as questões que envolvem controle de acesso em ambientes de computação em nuvem, onde prestadores de serviços e usuários precisam ser tratados de forma diferente em relação ao acesso às informações, sem causar um grande impacto ou complexidade na configuração de seus papéis. O resultado é um modelo de controle de acesso sensível ao contexto para a aplicação de forma pró-ativa do nível de acesso ao recurso, baseado em processamento de ontologia e método de análise semântica.

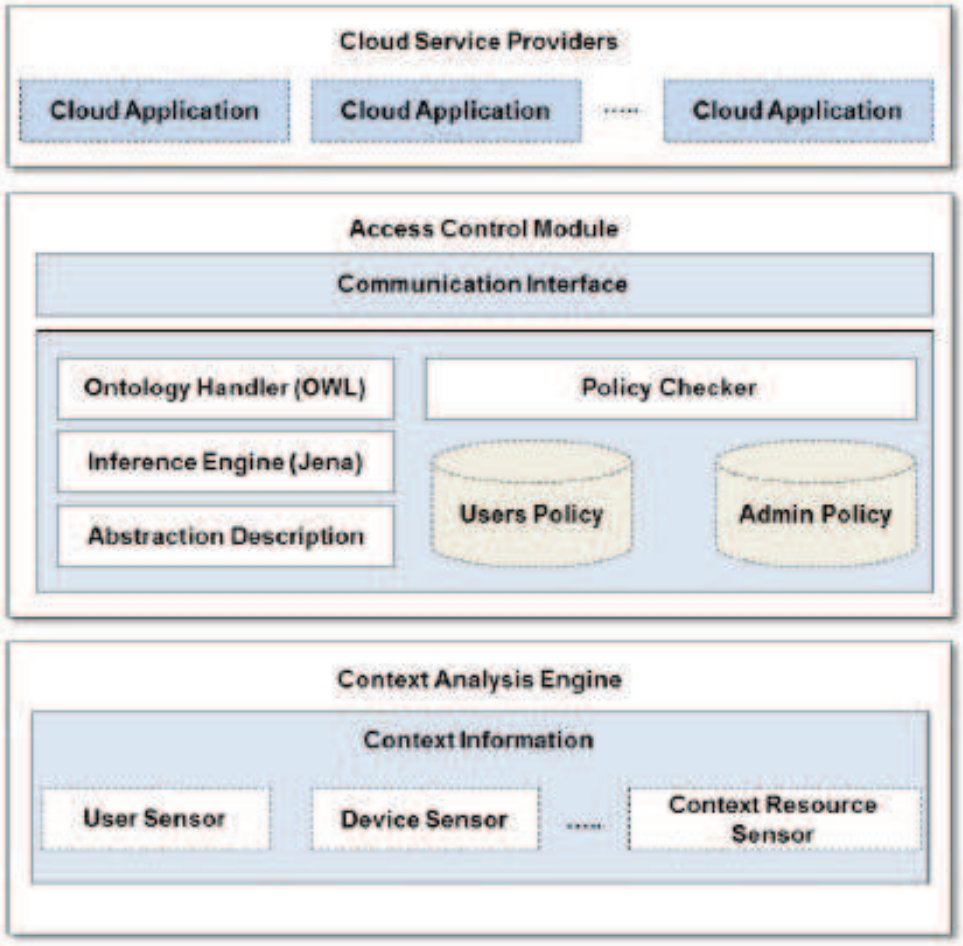
O modelo propõe satisfazer requisitos para ambientes de computação em nuvem, como:

- Papel do usuário pode ser dinamicamente e parcialmente delegado, alterando suas permissões.
- Restrições sobre o papel autorizado podem ser consideradas para controle de acesso dinâmico.
- Objetos, condições e obrigações para o acesso aos dados pode ser considerados, a fim de proteger informações no banco de dados.
- Acesso aos dados, se necessário pode ser rejeitado.
- Controle de acesso pode ser fornecido com base na localização e requisitos de equipamentos.
- O fator mais importante é a prevenção de qualquer utilização indevida dos direitos de acesso.

O trabalho apresenta um mecanismo para para proteção de aplicativos e sistemas, considerando os requisitos acima com base em tecnologias sensíveis ao contexto do ambiente de computação em nuvem.

A figura 15 mostra a arquitetura proposta. Observamos que Onto-ACM consiste de um mecanismo de análise de contexto que executa a seleção, análise, integração e disponibilização das informações sensíveis ao contexto, utilizando serviços específicos para este fim e, um módulo de controle de acesso que executa a função de gerenciamento e composição da política de segurança, apresentando uma interface de comunicação para este propósito.

Figura 15 – Arquitetura Onto-ACM



Fonte: (CHOI; CHOI; KIM, 2014)

O módulo de controle de acesso requer uma política de segurança e informações de contexto para autenticação e controle de acesso de um usuário com base no pedido de acesso deste usuário. O módulo de controle de acesso fornece a política de segurança e controle de acesso relacionado à informação de contexto. Finalmente, o processamento da ontologia baseia-se na integração das informações de contexto no módulo de controle de acesso.

O mecanismo de análise de contexto permite o acesso ao sistema de acordo com a política de segurança e com as condições de contexto. O manipulador de ontologia fornece a localização de todos os recursos que podem ser acessados baseando-se no papel do usuário e nas informações de contexto. Este método limita o acesso a recursos através da política de acesso no ambiente de computação em nuvem.

Serviços de computação em nuvem, geralmente oferecidos por prestadores de serviço, devem ser diferenciados de usuários com relação aos direitos de acesso. A camada de gerenciamento de informações controla a autenticação e direitos de acesso através de processamento de condições de ontologias e políticas de segurança, considerando a origem do acesso (usuário ou serviço).

Onto-ACM apresenta um modelo de controle de acesso que executa processamento de contexto utilizando o contexto, o nível de permissão, a condição da permissão, o propósito e as políticas de segurança para usuários e administradores. O processamento de inferência baseia-se no mecanismo de inferência JENA e a execução de consultas utiliza SPARQL. A principal vantagem deste modelo com relação a um modelo C-RBAC (Role-Based orientando ao contexto) é a facilidade da gestão das políticas de segurança, permitindo delegação de papéis de administrador e usuário, além disso, o modelo representa um controle de acesso dinâmico detalhado que pode resolver as questões características de computação em nuvem.

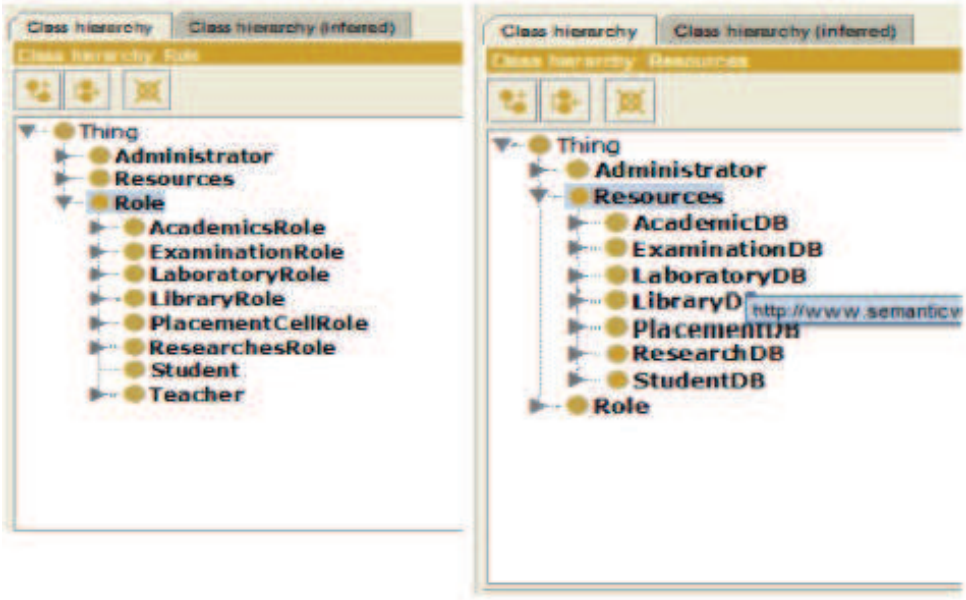
3.2 Authentication and Authorization: Domain Specific Role Based Access Control Using Ontology

Em (KATAL et al., 2013) encontramos um modelo completo de autenticação e autorização baseado em ontologia. O trabalho implementa RBAC (Role-Based Access Control) em um domínio específico, neste caso uma universidade, utilizando uma ontologia para representação do acesso. Os papéis são apresentados em forma de classes da ontologia, e permissões são associadas a estas classes. O acesso é realizado em duas etapas, autenticação e autorização. Os principais componentes do sistema são:

- Usuário
- Autenticação e atribuição de conjunto de papéis
- Atribuição de papéis
- Base de conhecimento

O usuário é um funcionário da universidade, com um papel específico a desempenhar (conjunto de permissões). Na autenticação, realizada com a utilização de ID e senha, caso o ID e senha sejam do administrador, ele poderá selecionar com qual papel irá atuar no sistema. Usuários que não são administradores recebem as permissões de acesso associadas ao seu papel específico. Na base de conhecimento são armazenados os dados de usuários, classes de da ontologia de acesso e o conjunto de permissões associados a cada papel existente. Na ontologia concebida (figura 16), existe uma classe principal *Role* (papel) e todos os diferentes papéis na universidade são as subclasses desta classe principal. Os recursos ou objetos para os quais o controle de acesso é destinado, também estão presentes sob a forma de classes. Estas duas classes estão ligadas com as propriedades *canAccess*, *canRead*, *canDelete* e outras, configurando assim

Figura 16 – Papéis da Ontologia de Acesso RBAC



Fonte: (KATAL et al., 2013)

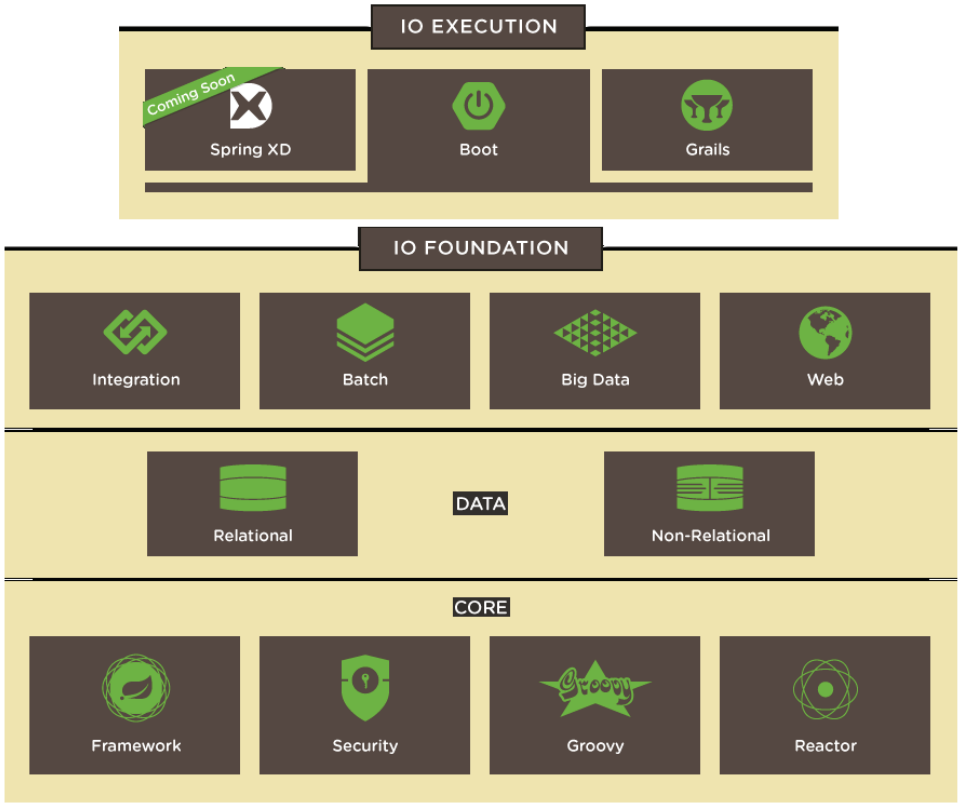
as permissões de acesso de cada papel existente. Esta estrutura facilita e simplifica a tarefa de atribuição de privilégios individuais a usuários, conforme o modelo RBAC, uma vez que é necessário apenas associar papéis já definidos a novos usuários ou a alteração de papéis de usuários já existentes. Os autores finalizam o trabalho apontando como possíveis trabalhos futuros a utilização de técnicas de auditoria de papel, com a finalidade de realizar transferências de papéis de acordo com ações realizadas, definição de um esquema de delegação de papéis (típicos de modelos RBAC) e autenticação externa com espelhamento de domínios, possibilitando o acesso de usuários externos à organização.

3.3 Spring Security

Spring Security (Spring Security, 2014) é um *framework* customizável para autenticação e controle de acesso. Faz parte da plataforma Spring IO, que constitui um conjunto de API's para desenvolvimento de aplicações. Spring IO possui código aberto e modular, permitindo a utilização customizada de cada uma das API's de acordo com a necessidade. De acordo com a Figura 17 podemos observar que o módulo *Security* encontra-se na camada Core da plataforma. Esta camada é composta por um conjunto de componentes de tempo de execução que trabalham em conjunto, dentro de cada camada.

Os serviços de segurança oferecidos pelo Spring Security são baseados em Java EE com ênfase especial a projetos criados com o Spring Framework, utilizando os mesmos princípios de injeção e dependência da plataforma. A utilização do Spring Security tem por objetivo suprir as necessidades de autenticação e autorização (controle de acesso) de um aplicativo qualquer. A autenticação no Spring Security suporta uma ampla gama de modelos. A maioria destes

Figura 17 – Camadas da Plataforma Spring



Fonte: http://spring.io/platform

modelos de autenticação são prestados por terceiros, ou são desenvolvidas por órgãos competentes, como a IETF. Além disso, o Spring Security fornece seu próprio conjunto de recursos de autenticação. Spring Security atualmente suporta a integração de autenticação as tecnologias:

- Cabeçalhos de autenticação HTTP BASIC (padrão IETF RFC-based)
- Cabeçalhos de autenticação HTTP Digest (padrão IETF RFC-based)
- Troca de certificados cliente HTTP X.509 (padrão IETF RFC-based)
- LDAP (uma abordagem muito comum de autenticação entre plataformas)
- Autenticação Form-based (para as necessidades de interface de usuário simples)
- Autenticação OpenID
- Autenticação baseada em cabeçalhos de solicitação de pré-estabelecidos
- Serviço de Autenticação Central JA-SIG (também conhecido como CAS, que é um sistema código aberto popular simples)
- Autenticação transparente de propagação de contexto para RMI e HttpInvoker (um protocolo de comunicação remota Spring)

- Autenticação automatica "remember-me"
- Autenticação anônima (permitindo que chamadas não autenticadas assumam automaticamente uma identidade de segurança especial)
- Autenticação Run-as (utilizada quando uma chamada deve utilizar uma identidade de segurança diferente)
- Java Authentication and Authorization Service (JAAS)

Spring Security permite ainda que se escreva um mecanismo de autenticação próprio, independente dos padrões citados. A autorização com Spring Security atende a três principais áreas de interesse:

- Autorização de requisições web;
- Autorização de métodos; e
- Autorização de acesso a instâncias de objetos de domínio individuais.

De posse de um objeto seguro (usuário ou dispositivo autenticado), retornado pela autenticação utilizada, e uma lista de atributos de metadados de segurança que se aplicam ao objeto (como uma lista de funções que são necessárias para o acesso ser concedido), uma interface do Springe Security, chamada *AccessDecisionManager*, verifica a autorização solicitada.

3.4 OpenLDAP

OpenLDAP (OpenLDAP, 2014) é uma implementação open source do LDAP (Lightweight Directory Access Protocol). Como um protocolo de aplicação aberto e padrão para acessar e manter serviços de informação de diretórios distribuídos através de uma rede IP (Internet Protocol), o LDAP desempenha um papel importante no desenvolvimento de aplicativos de intranet e Internet, permitindo o compartilhamento de informações sobre os usuários, sistemas, redes, serviços e aplicações em toda a rede. O OpenLDAP é independente de sistema operacional. Várias distribuições Linux incluem o pacote do OpenLDAP. O software também é executado nos sistemas operacionais BSD, AIX, HP-UX, Mac OS X, Solaris, Microsoft Windows (2000, XP, 2003, 2008, Vista, Windows 7 e 8) e z/OS. O OpenLDAP foi desenvolvido inicialmente pela Universidade de Michigan com as seguintes características principais:

- Suporte a IPv4 e IPv6;
- Autenticação (Cryrus Sasl-Kerberos V, GSSAPI, Digest-MD5);
- Segurança no transporte – SSL e TLS;
- Controle de acessos;

- Escolha entre banco de dados;
- Capacidade de atender a múltiplos bancos de dados simultaneamente;
- Alta performance em múltiplas chamadas;
- Replicação de base;

O OpenLDAP é composto pelos seguintes elementos:

- SLAPD (Stand Alone LDAP Daemon) – Servidor;
- Bibliotecas diversas de implementação do protocolo LDAP;
- Ferramentas, utilitários diversos e clientes de exemplo;

Possui basicamente 2 arquivos de configuração, que são, respectivamente:

- slapd.conf – configuração do serviço;
- ldap.conf – configuração para acesso dos clientes à base;

O OpenLDAP possui arquivos padrões chamados de *schemas*. Para (SUNGAILA, 2007), um *schema* é um conjunto de regras que define atributos, classes de objetos, e controles indicando onde cada dado pode ser armazenado. Os *schemas* permitem manter a consistência dos dados. Uma importante característica desses arquivos é serem extensíveis e assim pode-se adicionar mais atributos ou classes em função das necessidades. Para usar um novo *schema* é necessário incluí-lo no arquivo de configuração slapd.conf.

Os *schemas* em OpenLDAP definem:

- Quais as classes de objetos podem ser inseridas num diretório;
- Quais os atributos de uma determinada classe de objetos;
- Os valores possíveis para os atributos;

Se um objeto (entrada), não obedecer às regras do *schema*, este não pode ser inserido no diretório. Portanto cada entrada estará condicionada a uma hierarquia de armazenamento dos dados na base LDAP.

Para implementação de um estudo de caso foi utilizado o OpenLDAP e o conjunto de classes fornecido pela Novell (NOVELL, 2013), o que permitiu o acesso e o gerenciamento das informações do OpenLDAP.

Este conjunto de classes foi utilizado por estarem baseados no programa de interfaces para aplicações Java do IETF e por permitir a utilização da linguagem Java, utilizada na maioria dos projetos da plataforma MILOS.

3.5 Análise

Os trabalhos analisados possuem diferentes características com relação a porte, flexibilidade ou nível de personalização e arquitetura. A tabela 2 apresenta um comparativo das características relevantes para o presente trabalho.

Tabela 2 – Comparativo dos Trabalhos Relacionados

Ferramenta	**Características**					
	Autenticação	Autorização	Agentes	Base Semântica	Federado	Acesso Sem Sessão
Onto-ACM	ND	Sim	ND	Sim	*2	*2
AADS-O	ND	Sim	ND	Sim	*2	*2
Spring	Sim	*1	Não	*1	Sim	*1
OpenLDAP	Sim	Sim	Não	Não	Sim	*1
Fayol	Sim	Sim	Sim	Sim	Sim	Sim

Fonte: Elaborada pelo autor.

- ND - Não Definido
- *1 - Possível com algum esforço de implementação
- *2 - Implementação possível se utilizar a arquitetura Fayol com uso de agentes

Spring Security e OpenLDAP representam soluções robustas e abrangentes para autenticação e autorização. Observamos porém, que a autorização é um processo estreitamente relacionado ao ambiente no qual será utilizado. Ambos *frameworks* exigiriam esforço de adaptação à plataforma MILOS de igual ou maior impacto que a solução adotada. Seriam utilizados, basicamente como uma ferramenta poderosa e flexível para autenticação de usuários, o que neste caso é o mais simples deles, de um único fator, ou seja, ID e senha. Além disso, o *framework* adotado deveria estar replicado na plataforma MILOS de modo a não permanecer dependente de projetos externos.

Os artigos apresentados, utilizando ontologias, aproximam-se neste aspecto, da plataforma MILOS. Observamos que a utilização de ontologias para definição de acessos, classes e permissões obteve resultados positivos nos trabalhos estudados. Observamos também que o modelo RBAC, amplamente aceito e utilizado, tanto no meio acadêmico quanto no meio empresarial, pode ser modelado com vantagem utilizando ontologias. Algoritmos para a realização da autenticação e autorização podem ser implementados com o uso de agentes de software, tornando o modelo totalmente compatível com a plataforma MILOS.

A pesquisa realizada demonstrou que o uso de ontologias em mecanismos de controle de acesso é bastante recente e apresenta evidentes vantagens em relação aos métodos tradicionais por introduzir a possibilidade de processamento e inferências nas ontologias de acesso, com o objetivo de resolver questões tanto de autenticação quanto de autorização. A coleta de dados

adicionais referentes ao contexto e localização dos agentes, conforme o trabalho apresentado em 3.1, permite facilitar o gerenciamento do controle de acesso em ambientes de computação em nuvem ou mesmo em redes locais porém com grande diversidade de equipamentos.

Entretanto, dois aspectos importantes que estão sendo considerados no presente trabalho, mas que não foram observados nos artigos que apresentam modelos ontológicos para os serviços de autenticação e autorização é a falta de mecanismos e protocolos que permitam construir federações destes serviços, além de um cuidado com a segurança nos canais de comunicação entre os serviços, ou do serviço com seus clientes. Embora não tenham sido objetivos específicos dos artigos analisados que apresentam o OpenLDAP e Spring Security, é importante notar que as soluções mais tradicionais de autenticação e autorização podem trabalhar interligadas em federação, além de suportar mecanismos de segurança.

Em termos gerais, a análise do estado da arte permitiu observar que a criação de um modelo ontológico de autenticação e controle de acesso utilizando o modelo RBAC, armazenada em base local utilizando o banco já existente na plataforma MILOS, em conjunto com a implementação de mecanismos para controle de acesso através de agentes de software que podem operar tanto em forma local quanto federada, representa a estratégia ideal para atingir os objetivos definidos neste trabalho, além de permitir avançar as pesquisas dessa nova abordagem para a autenticação e autorização.

4 FAYOL: UM SISTEMA DE GERÊNCIA E CONTROLE DE ACESSO À PLATAFORMA MILOS

Neste capítulo será apresentado o sistema *Fayol* como uma solução de gerência e controle de acesso à plataforma MILOS, que pretende oferecer funcionalidades observadas nos trabalhos apresentados no capítulo anterior, com evolução no sentido de compatibilidade e integração à plataforma MILOS, representando seu modelo e arquitetura, uma solução abstrata, formal e independente.

O sistema *Fayol* implementa a gerência de usuários, perfis de usuários, aplicações, perfis de aplicações e recursos utilizados na plataforma MILOS em conjunto com o controle de acesso a esta plataforma. Para isso, utiliza uma base de dados ontológica (SILVA; GLUZ, 2012), suportada por uma camada TDB de JENA (JENA, 2013). Esta base ontológica, que já armazenava os metadados dos objetos de aprendizagem contidos na plataforma MILOS, foi estendida pela incorporação da ontologia de acesso e instâncias (indivíduos) desta ontologia. A base de dados TDB poderá ser distribuída em um ou mais servidores, de acordo com a necessidade da plataforma.

4.1 Especificação de Requisitos

Conforme citado anteriormente, foi utilizado Tropos para a especificação de requisitos. De acordo com (CASTRO; ALENCAR; SILVA, 2006), a metodologia Tropos define cinco fases do desenvolvimento de software, conforme apresentado na Figura 14, com o objetivo de descrever os elementos relevantes (atores, objetivos e dependências) para a especificação formal de um domínio e os relacionamentos entre eles.

4.1.1 Requisitos Iniciais

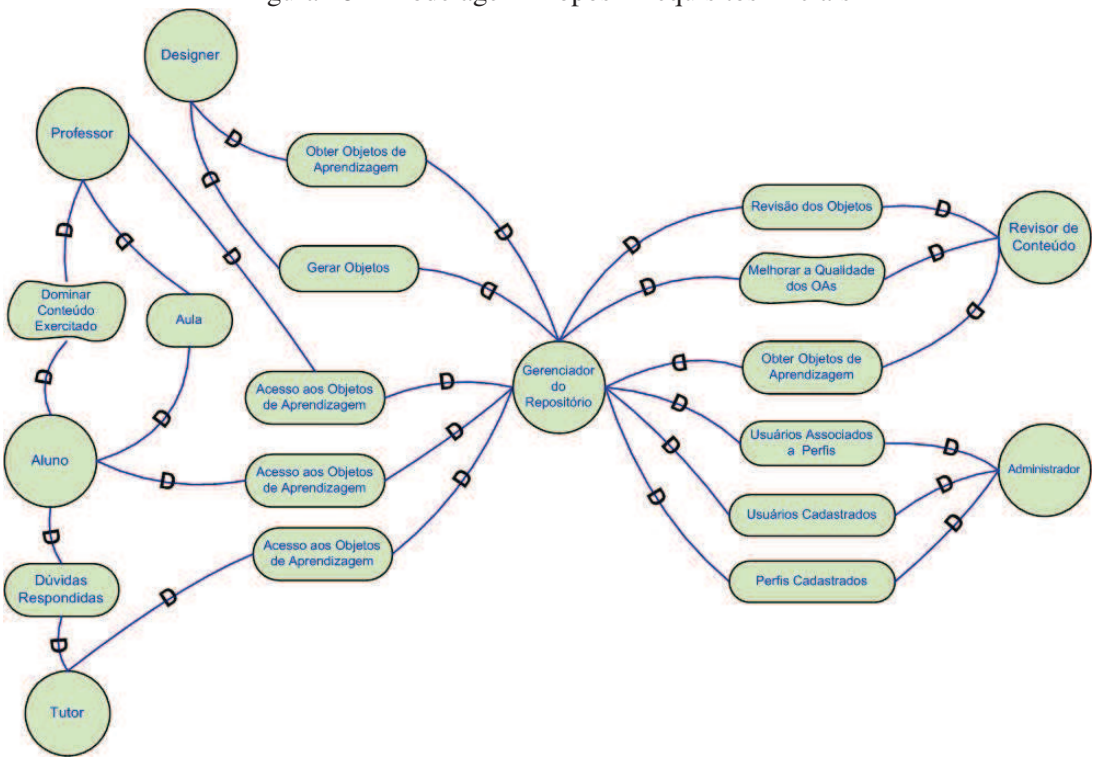
Nesta fase foram identificados os principais atores Aluno, Professor, Tutor, Designer, Administrador, Revisor de Conteúdo e o agente Gerenciador do Repositório, conforme diagrama apresentado na Figura 18. Observamos o agente gerenciador como figura central do diagrama e as principais metas relacionadas são: Acesso aos objetos de aprendizagem, criação e revisão destes objetos e usuários cadastrados e associados aos perfis adequados.

4.1.2 Requisitos Finais

Nesta fase foram identificadas as tarefas, metas, recursos e dependências internas a cada domínio de ator especificado na fase anterior. O diagrama refinado desta etapa está representado nas Figuras 19 e 20.

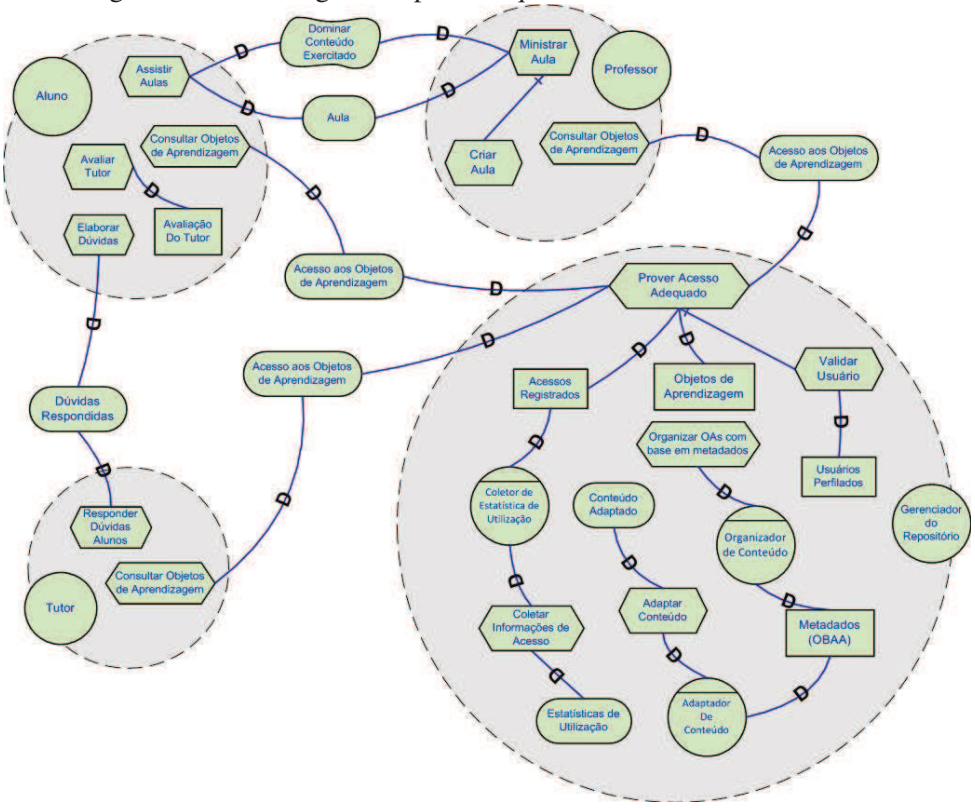
Observamos que as metas comuns entre os atores estão representadas fora do domínio do

Figura 18 – Modelagem Tropos - Requisitos Iniciais



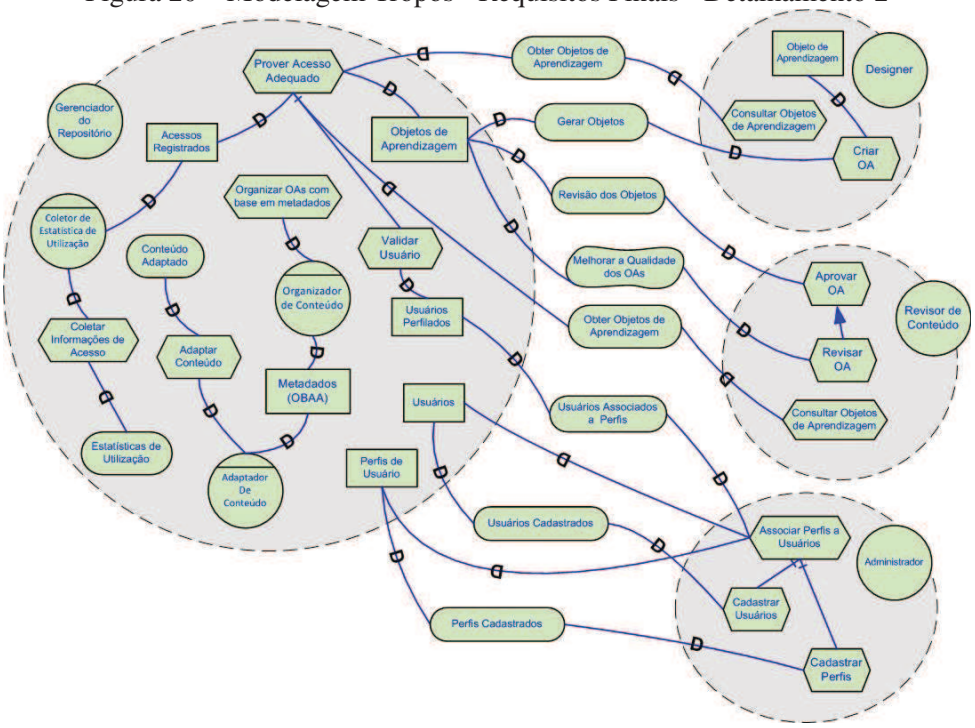
Fonte: Elaborada pelo autor

Figura 19 – Modelagem Tropos - Requisitos Finais - Detalhamento 1



Fonte: Elaborada pelo autor

Figura 20 – Modelagem Tropos - Requisitos Finais - Detalhamento 2



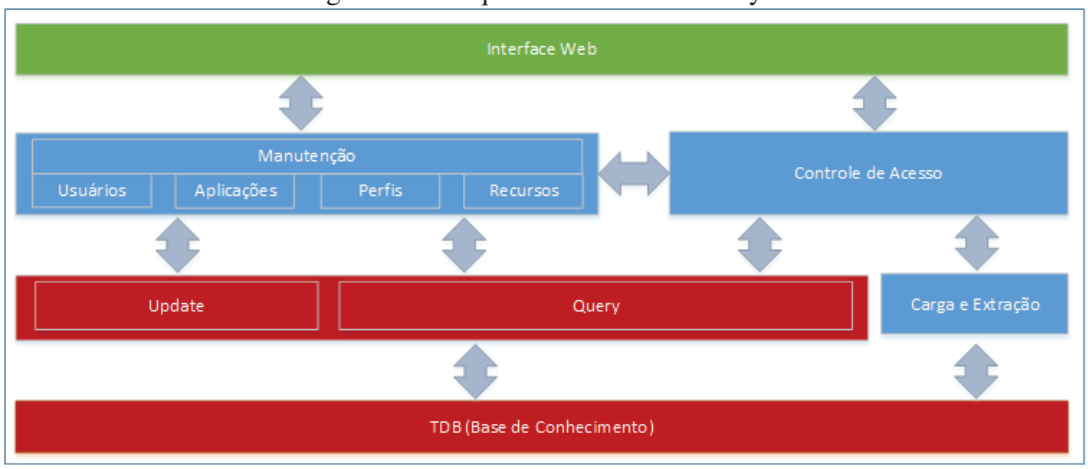
Fonte: Elaborada pelo autor

ator. Metas, Recursos, Tarefas e demais componentes que foram especificados para cada Ator são representados no domínio de cada Ator.

4.2 Arquitetura

A figura 21 apresenta a arquitetura definida para o sistema *Fayol*. A estrutura apresenta as camadas onde os agentes e interfaces são implementados. As tarefas de cada camada são executadas por um ou mais agentes, conforme a necessidade de processamento.

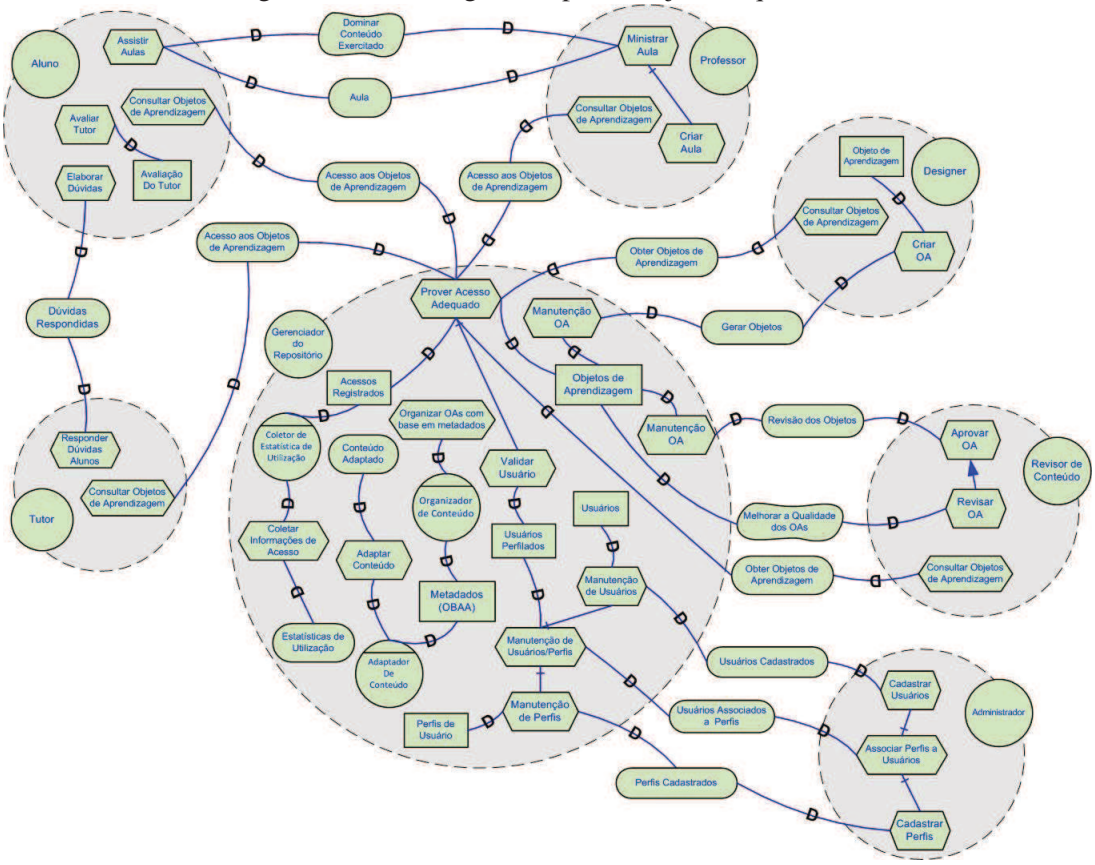
Figura 21 – Arquitetura do Sistema Fayol



Fonte: Elaborada pelo autor

A especificação de requisitos arquiteturais, representada na Figura 22 por um diagrama SD, descreve a estrutura multi-agente interna do ator Gerência do Repositório (que representa o sistema *Fayol*) e as relações de dependência destes agentes com os demais atores que irão interagir com este sistema. Esta especificação permite inferir como os componentes do sistema interagem como um todo, buscando atingir suas metas. Em Tropos, os estilos arquiteturais organizacionais para aplicações cooperativas, dinâmicas e distribuídas estão fundamentados de forma a atender um sistema multi-agente. Assim os conceitos fundamentais são intencionais e sociais e por esta razão está voltado a teorias que estudam estruturas sociais. Algumas tarefas definidas foram distribuídas entre diferentes agentes de software na implementação do modelo. Os agentes desempenham uma ou mais tarefas do diagrama.

Figura 22 – Modelagem Tropos - Projeto Arquitetural



Fonte: Elaborada pelo autor

Conceitos de teoria organizacional formam então a base para o estilo arquitetural de sistemas multi-agente, onde podemos situar o Modelo de Dependência Estratégica (SD), o Modelo de Razão Estratégica (SR), entradas para a próxima etapa, o Projeto Detalhado.

4.2.1 Projeto Arquitetural e Dinâmica do Sistema em UML

Por UML ser uma linguagem visual de uso geral para especificar, construir e documentar artefatos de sistemas, com possibilidade de utilização em todos os domínios de aplicações

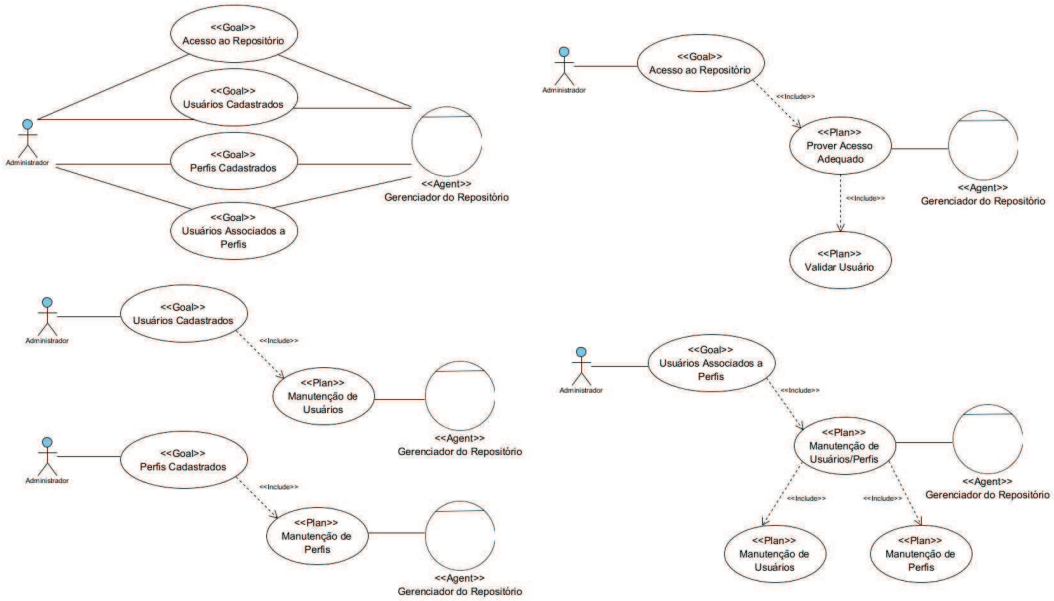
e plataformas de implementação, incluindo sistemas multi-agente, selecionamos os seguintes diagramas de UML para complementar a especificação do sistema *Fayol*:

- Use Cases
- Diagrama de Comunicação
- Diagrama de Sequência

4.2.1.1 Use Cases

Para cada relação de dependência do diagrama Tropos, criou-se um Use Case específico para representar esta relação de dependência. A Figura 23 mostra a relação de dependência do Ator - Administrador com o Agente – Gerenciador de Repositório e sua representação utilizando um diagrama Use Case. As metas do diagrama Tropos, comuns ao Administrador e

Figura 23 – Use Case: Administrador – Gerenciador de Repositório



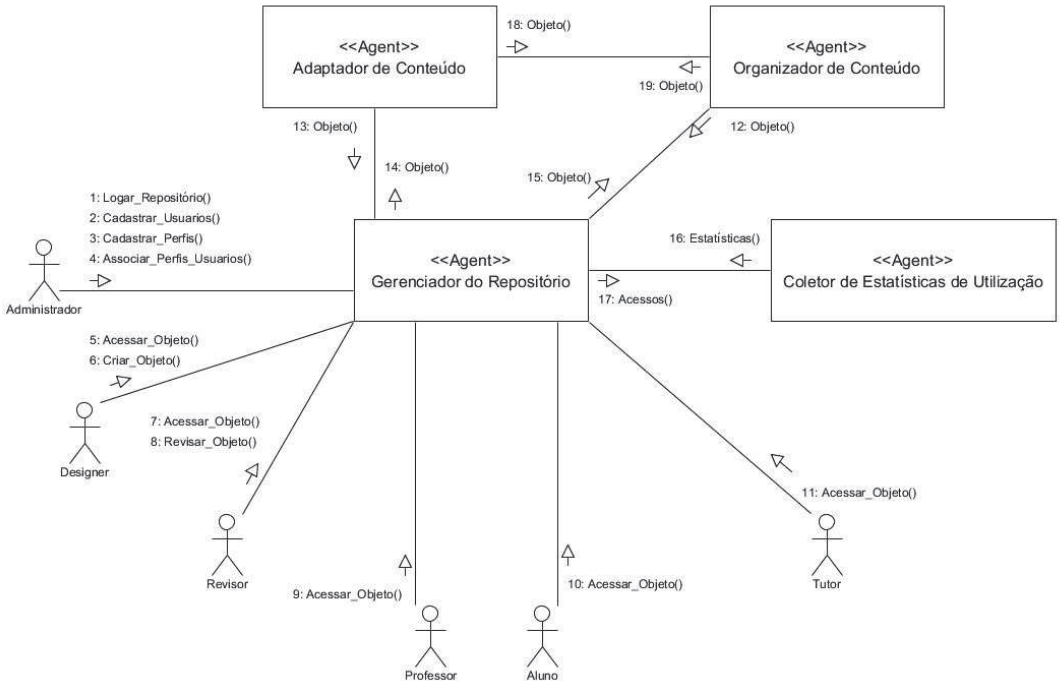
Fonte: Elaborada pelo autor

ao Gerenciador de Repositório, requerem Planos para serem atingidas. As tarefas e sub-tarefas existentes no diagrama Tropos são representadas como Planos no Use Case. Assim como em Tropos existe a possibilidade de uma tarefa estar associada a tarefas complementares ou sub-tarefas, no Use Case também teremos esta representação com Planos que incluem Sub-Planos ou Planos complementares ao Plano principal para atingir uma Meta.

4.2.1.2 Diagrama de Comunicação

O diagrama de comunicação representa a troca de mensagens entre os agentes/atores representados. Através deste diagrama, podemos observar de forma visual as diversas mensagens trocadas pelos componentes do sistema. A Figura 24 apresenta o diagrama de comunicação obtido na especificação.

Figura 24 – Diagrama de Comunicação ou Colaboração (Geral)



Fonte: Elaborada pelo autor

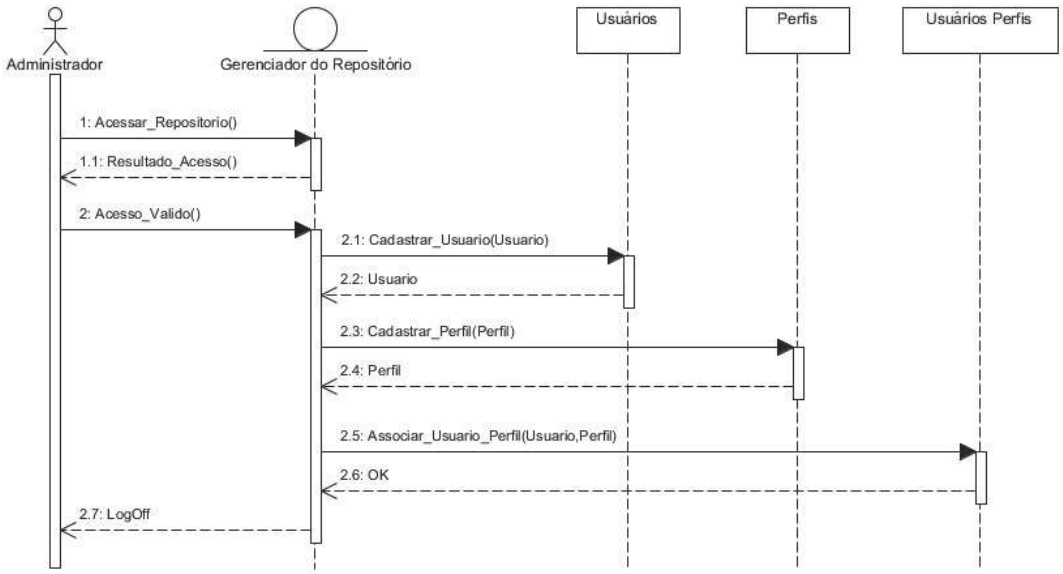
4.2.1.3 Diagrama de Sequência

Semelhante ao diagrama de comunicação/colaboração, o diagrama de sequência permite a visualização da ordem com que a troca de mensagens ocorre entre os agentes/atores. Como as mensagens são detalhadas e ordenadas, é interessante o agrupamento de diagramas por relação de dependência o que torna mais simples e mais legível a interpretação. A Figura 25 mostra o diagrama de sequência da relação Administrador <-> Gerenciador de Repositório.

4.2.2 Persistência dos Dados

Parte integrante do *framework* JENA, o TDB foi utilizado para a persistência de dados e metadados no formato RDF, estendendo o banco atualmente utilizado pelo MSSearch (SILVA; GLUZ, 2012) na plataforma MILOS. Todo o acesso de aplicações ao banco de dados, à exceção

Figura 25 – Diagrama de Sequência Administrador – Gerenciador de Repositório



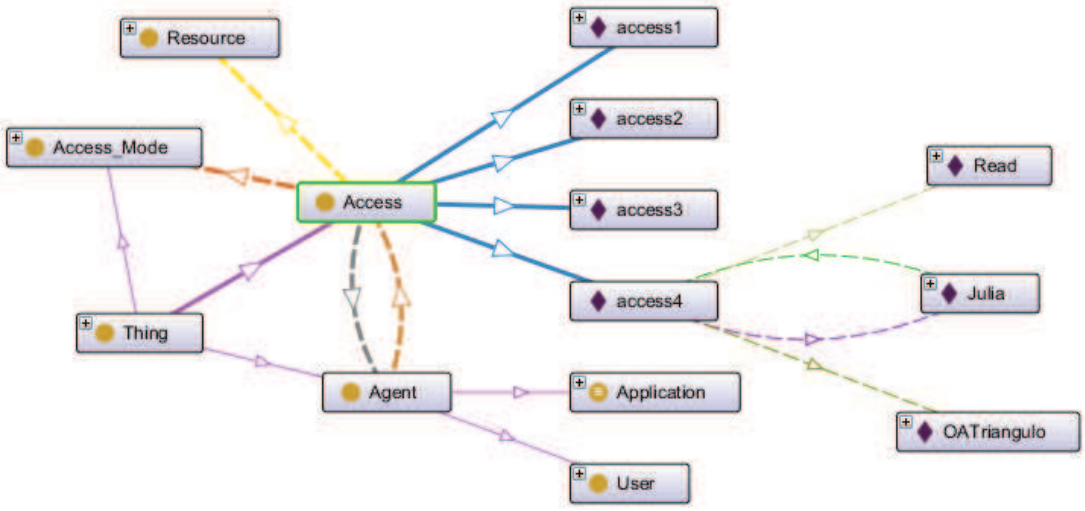
Fonte: Elaborada pelo autor

de cargas e extrações, por razões detalhadas posteriormente, é realizado através dos agentes *Query* e *Update* criados para este fim.

4.2.2.1 Ontologia de Acesso

Foi criada uma ontologia de acesso para modelar os agentes, recursos e modos de acesso que serão utilizados no sistema *Fayol*. Na figura 26 apresentamos a ontologia. A classe *Access*

Figura 26 – Ontologia de Acesso Fayol



Fonte: Elaborada pelo autor

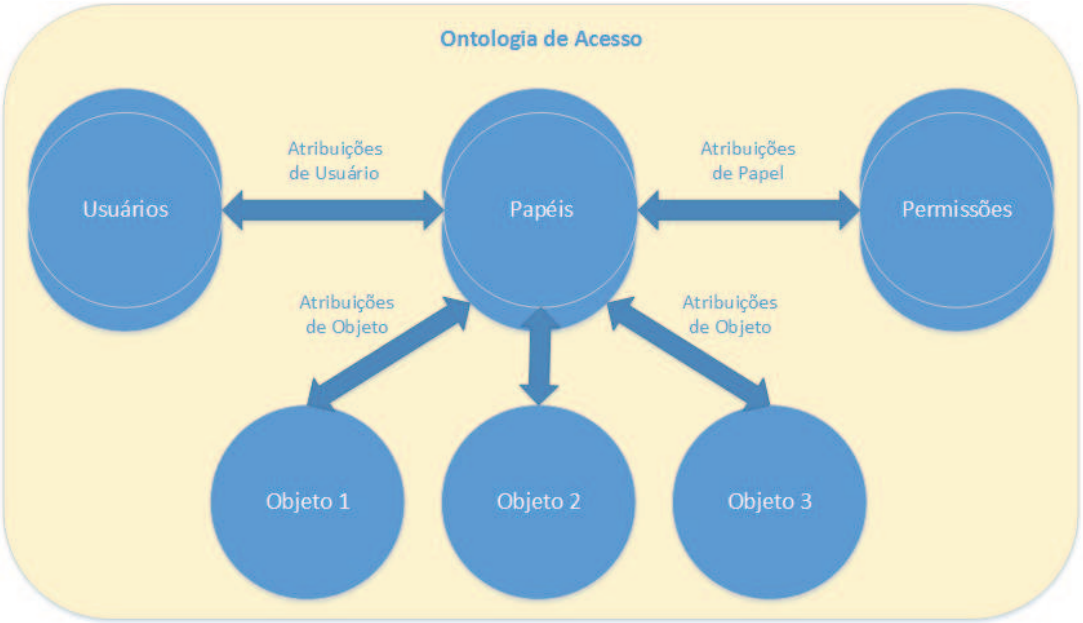
realiza o relacionamento entre um agente usuário, um recurso e um modo de acesso permitido. O usuário poderá ser um *User* ou uma *Application*, um recurso poderá ser um *Learning Object*

ou uma *Application As Resource* quando desejarmos tratar uma aplicação da plataforma como um objeto que poderá ou não ser executada. Os modos de acesso foram definidos como *Control, Read, Write, Delete e Execute*.

Na figura 26 observamos que a instância *access4*, indica que o usuário *Júlia* possui acesso *Read* ao objeto de aprendizagem *OATriangulo*.

Podemos afirmar que a ontologia criada para o *Fayol* implementa um modelo RBAC modificado, conforme apresentado na Figura 27.

Figura 27 – Modelo RBAC Fayol



Fonte: Elaborada pelo autor

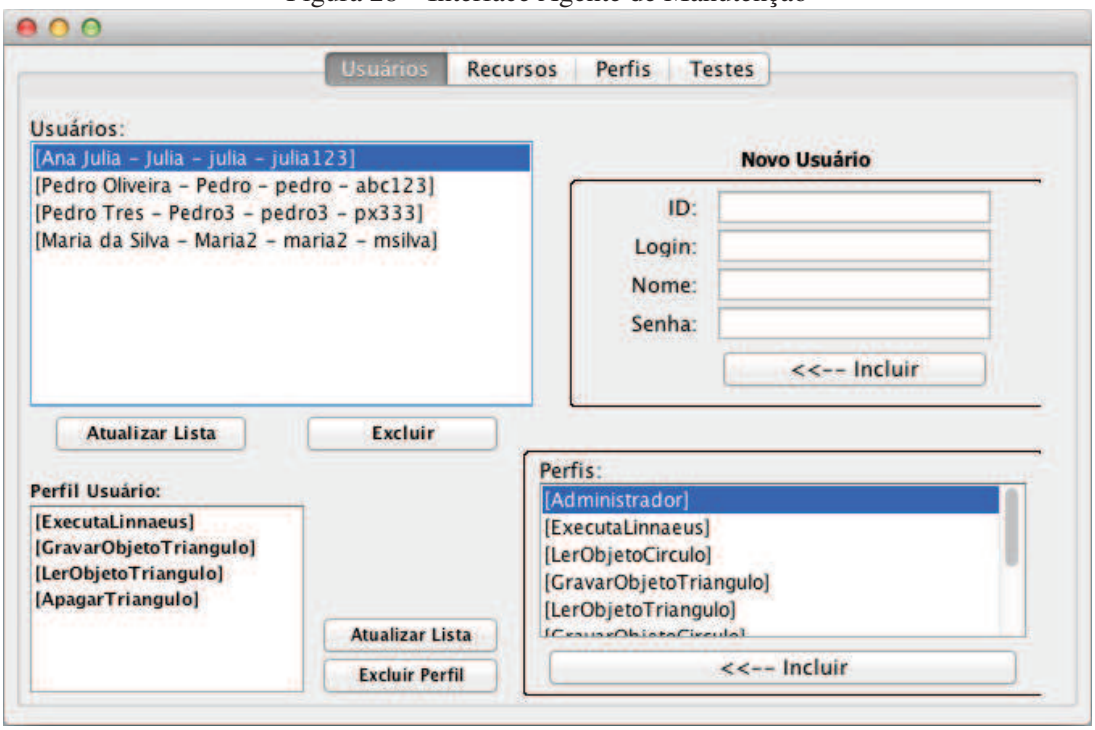
4.2.2.2 Interface Web

Esta camada será responsável pela apresentação e interação com o usuário. Seu desenvolvimento é em Java, utilizando Tomcat como servidor de aplicação, mantendo-se assim o padrão de compatibilidade da plataforma MILOS existente.

4.2.2.3 Manutenção e Controle de Acesso

Nesta camada encontramos os agentes de manutenção e controle de acesso. O agente de manutenção é responsável, como o próprio nome diz, pelas operações de inclusão, alteração e exclusão de dados de usuários, perfis de usuários, aplicações, perfis de aplicações e recursos. Para isso dispõe de uma interface gráfica amigável (figura 28) para sua operação. O acesso ao agente de manutenção é restrito ao administrador da plataforma, com ID e senha específicos para este fim.

Figura 28 – Interface Agente de Manutenção



Fonte: Elaborada pelo autor

O agente de controle de acesso realizará a verificação de acesso de um determinado usuário ou aplicação e a funcionalidade ou mesmo a aplicação requisitada, no caso de usuário, que deseja acessar, retornando o acesso ou não de acordo com o perfil do requerente.

Assim como todos os demais agentes da plataforma, os agentes da camada de manutenção e controle de acesso fazem uso dos serviços oferecidos pelos agentes *Query* e *Update*, enviando comandos SPARQL diretamente a eles de acordo com a operação desejada.

O agente de controle de acesso poderá fornecer acesso ao agente de Carga e Extração quando for solicitado e permitido.

4.2.2.4 Camada Update, Query, Carga e Extração

Esta camada é responsável pelas tarefas desempenhadas pelos agentes *Query* e *Update*, integrantes do MSSearch (SILVA; GLUZ, 2012) e o agente de Carga e Extração. Estes agentes atuam como um *gateway* para o banco de dados TDB. Os agentes *Query* e *Update* recebem requisições dos demais agentes do ambiente, centralizando o acesso à camada de persistência. Como o próprio nome identifica, o agente *Query* realiza consultas e o agente *Update* atualizações e exclusões. A linguagem utilizada para este fim é SPARQL 1.1[1]. O agente *Update* foi modificado para suportar SPARQL-UPDATE, funcionalidade não existente na versão anterior.

O agente de Carga e Extração deverá ser utilizado em situações que envolvem movimenta-

[1] http://www.w3.org/TR/rdf-sparql-query/

ção de um grande volume de dados da base. Chamadas recorrentes aos agentes *Query* e *Update* para realizar esta tarefa acarretaria uma degradação da performance do sistema.

Estes agentes podem ser distribuídos em vários servidores no ambiente, assim como bases TDB, possibilitando escalonamento conforme necessidade.

5 EXPERIMENTOS DE AVALIAÇÃO

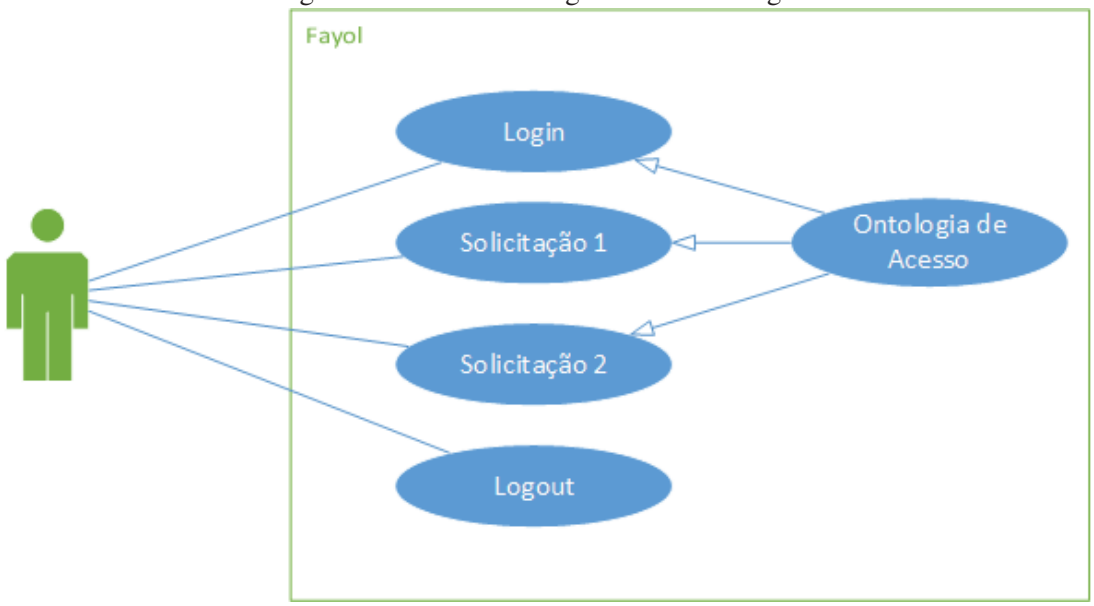
A metodologia para realização dos experimentos e sua validação, em relação aos objetivos propostos, foi baseada na criação de cenários de configurações (usuários, perfis e recursos) e casos de uso (ações de acesso), seguida da condução de experimentos em laboratório para verificar se o modelo apresentava o resultado esperado.

Além de verificar as propriedades funcionais do modelo e do mecanismo implementado de autenticação e autorização, também foi objetivo dos experimentos fazer uma avaliação preliminar do desempenho destes elementos, em conformidade com a prática de avaliação das soluções convencionais.

5.1 Experimento Login - Acesso - Logout

Neste cenário mais simples de teste, foram cadastrados usuários, perfis, recursos e a associação de perfis a usuários. O Use Case da figura 29 representa o experimento realizado.

Figura 29 – Use Case Login - Acesso - Logout



Fonte: Elaborada pelo autor

O usuário loga-se à plataforma fornecendo ID e senha corretos, não obtendo sucesso se informar incorretamente qualquer um dos parâmetros.

Uma vez logado e autenticado corretamente, o usuário (agente) recebe um *token* identificador que será válido durante toda sua sessão, e será utilizado quando solicitar qualquer acesso ao sistema. A figura 30 apresenta a interface criada para validação do sistema *Fayol* após a execução do login do usuário "Julia".

Verificou-se que o sistema respondia corretamente, de acordo com o perfil do usuário solicitante. As figuras 31 e 32 mostram o resultado de uma ação permitida e uma não permitida

Figura 30 – Validação de Login do Fayol

Fonte: Elaborada pelo autor

solicitadas pelo usuário "Julia". Consta no perfil deste usuário que ele pode executar a aplicação Linnaeus da plataforma MILOS.

A figura 31 apresenta o resultado "não permitido"quando da solicitação de execução da aplicação MSSearch. A figura 32 apresenta o resultado "permitido"quando da solicitação de execução da aplicação Linnaeus, que faz parte de um dos perfis do usuário.

Observou-se que ao realizar o logout, o sistema invalidava o *token* do usuário, não permitindo sua utilização, necessitando de uma nova ação de login para receber um novo *token* válido.

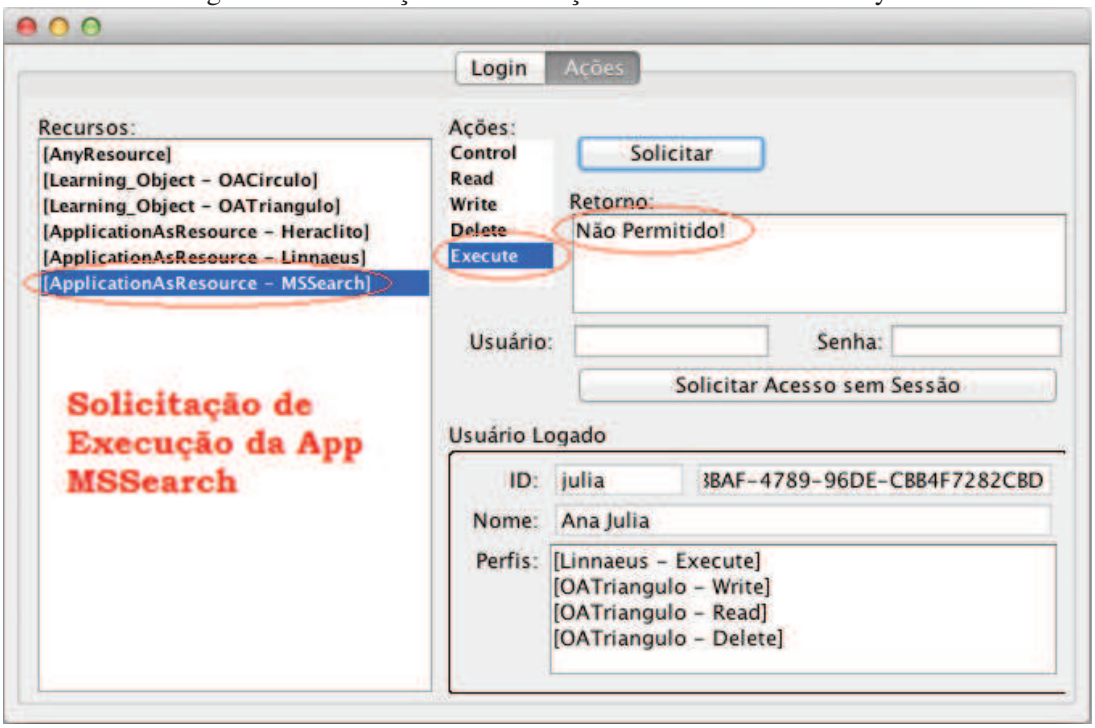
5.2 Experimento Login - Acesso - Logout (federado)

Neste cenário, semelhante ao experimento da sessão anterior, foram cadastrados usuários, perfis, recursos e a associação de perfis a usuários em um container remoto ao agente que recebia as solicitações do usuário. A figura 33 apresenta o cenário deste experimento.

Todas as solicitações são repassadas ao agente capaz de autenticar e autorizar o usuário, retornando a resposta às requisições ao ponto de origem. Verificou-se o mesmo comportamento esperado, conforme o experimento anterior, evidenciando que o gerenciamento e o controle de acesso pode se dar local ou remotamente, ficando ao encargo do sistema a busca do local onde determinado usuário pode ser autenticado e autorizado.

Nos casos em que um ID ou senha incorretos eram enviados, não conseguindo autenticar o usuário em nenhum dos agentes de controle da plataforma, a resposta de ID ou senha incorretos

Figura 31 – Validação de Solicitação "Não Permitida"do Fayol

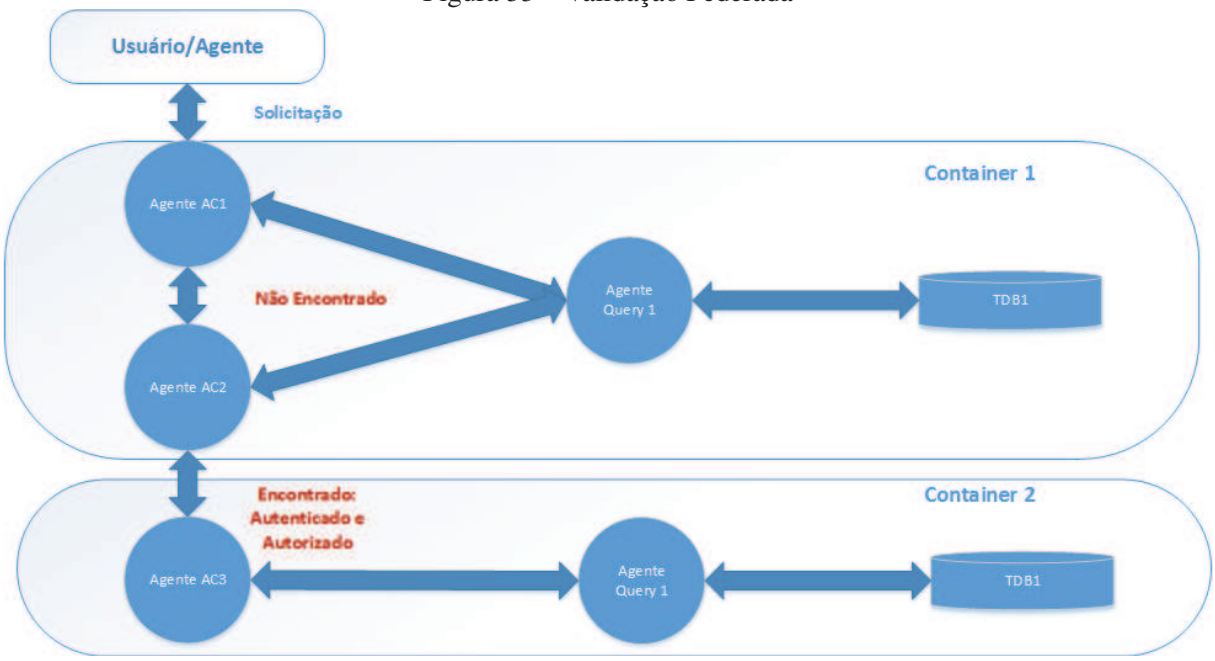


Fonte: Elaborada pelo autor

Figura 32 – Validação de Solicitação "Permitida"do Fayol

Fonte: Elaborada pelo autor

Figura 33 – Validação Federada



Fonte: Elaborada pelo autor

era retornada.

5.3 Experimento Acesso sem Sessão

Criou-se um cenário em que um agente necessitava executar uma aplicação da plataforma MILOS, sem que fosse necessário ou desejado o login deste agente ao sistema. A figura 34 apresenta o resultado deste experimento.

Verificou-se que o retorno do controle de acesso, estava de acordo com o perfil do agente solicitante, retornando corretamente se era permitido ou não a execução da aplicação desejada, sem a criação da sessão ou *token* para a realização da autorização.

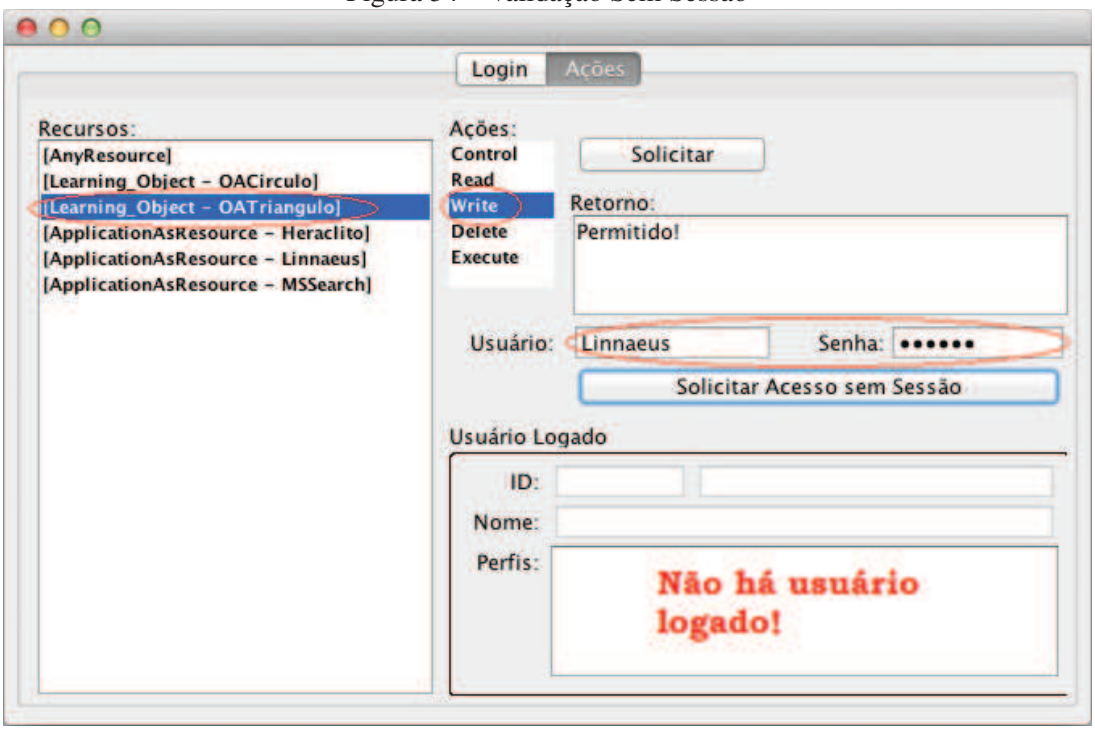
5.4 Avaliação do Desempenho

Embora não esteja diretamente relacionada aos objetivos do trabalho, a medição da performance do sistema visa prever o impacto no aumento de usuários e consequentemente de requisições de acesso, permitindo ter uma idéia da viabilidade de aplicação das novas tecnologias empregadas neste trabalho, em comparação com soluções convencionais.

Para realização destas medidas foi criada uma função no programa de testes, onde seleciona-se uma quantidade de acessos sucessivos de um determinado usuário, medindo o tempo para a execução destas requisições.

Foram utilizados lotes de execuções com as quantidades de 500, 1000, 2000, 4000 e 8000 acessos sucessivos para cada lote. As execuções foram realizadas para respostas esperadas de

Figura 34 – Validação Sem Sessão



Fonte: Elaborada pelo autor

"Acesso Permitido"e "Acesso não Permitido"separadamente. Também foram realizadas as mesmas quantidades de execuções para acessos com sessão (usuário logado) e sem sessão (usuário não logado).

Para cada lote de execução, foi iniciado um *container* local JADE e encerrado logo após a execução e medição, desta forma, *buffers* que eventualmente tenham sido criados, não interferiram nas execuções subsequentes. O resultado final é composto da média aritmética de três execuções de cada lote definido.

A tabela 3 apresenta as medições obtidas e a figura 35 apresenta um gráfico comparativo destas medições.

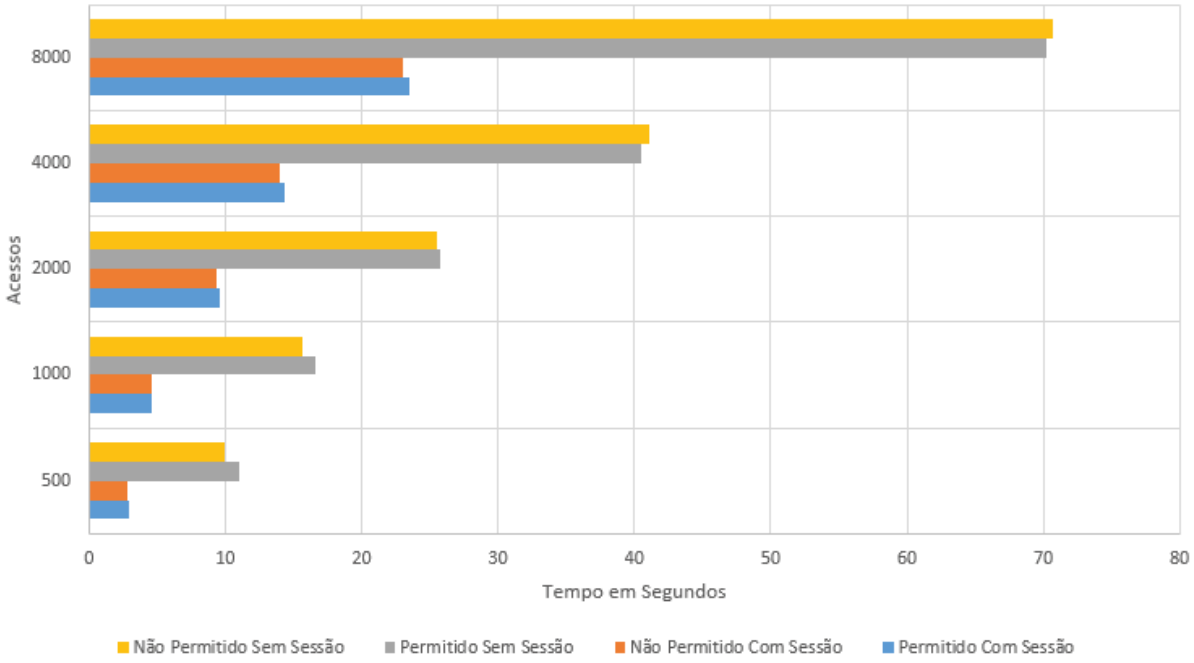
Tabela 3 – Medidas de Tempo em Segundos

	Com Uso de Sessão		**Sem Uso de Sessão**	
Acessos	**Permitido**	**Não Permitido**	**Permitido**	**Não Permitido**
500	2,94070	2,86527	11,00204	9,91324
1000	4,64319	4,59782	16,57985	15,66593
2000	9,54312	9,36408	25,80456	25,57108
4000	14,35791	14,03330	40,47753	41,06140
8000	23,48325	22,98405	70,17281	70,64062

Fonte: Elaborada pelo autor

Estes experimentos foram todos executados em uma máquina virtual com a seguinte configuração:

Figura 35 – Tempo médio para acessos



Fonte: Elaborado pelo autor

- 2 Processadores de 2 núcleos a 1.9 Gigahertz de frequência
- 4 Gigabytes de memória RAM
- Sistema Operacional Mac OS X Versão 10.9.4

A máquina virtual, por sua vez, está hospedada em um servidor com a seguinte configuração:

- 2 Processadores Intel Xeon E5-2420 de 6 núcleos a 1.9 Gigahertz de frequência
- 32 Gigabytes de memória RAM
- Sistema Operacional Windows Server 2012 de 64 bits

Os experimentos de controle de acesso das seções 5.1, 5.2 e 5.3, foram executados em notebooks e nas máquinas do laboratório do projeto OBAA-MILOS, basicamente com a configuração:

- 1 Processador Intel i5 de 4 núcleos a 3 Gigahertz de frequência
- 4 Gigabytes de memória RAM
- Sistema Operacional Linux Ubuntu Versão 11

5.5 Análise dos Experimentos

Verificamos que o sistema *Fayol* apresentou resultados positivos com relação aos objetivos propostos. Em todos os experimentos de verificação funcional os objetivos propostos foram atendidos. Com relação ao funcionamento integrado à plataforma MILOS, o sistema *Fayol* não produziu nenhum efeito indesejado, seja de queda de performance ou instabilidade, os agentes foram executados em containers JADE individuais ou em conjunto com agentes da aplicação MSSearch, utilizando bases TDB com a ontologia de acesso e com objetos de aprendizagem da plataforma OBAA-MILOS (VICARI; GLUZ, 2011).

O desempenho observado no experimento da seção 5.4 está de acordo com o esperado e é plenamente satisfatório com possibilidade de atender à demanda atual. Este experimento, em particular, permitiu verificar a escalabilidade do mecanismo implementado. A utilização de agentes sobre a plataforma de comunicação JADE permite sua distribuição em vários servidores, atendendo a situações onde o domínio de aplicação necessite atender um número elevado de solicitações do sistema.

Não encontramos, em nenhum dos trabalhos relacionados analisados no capítulo 3, resultados de testes de desempenho, o que não nos permitiu realizar um estudo comparativo direto entre o Fayol e estes trabalhos. É importante salientar, entretando, que OpenLDAP e outras soluções mais tradicionais certamente já foram exaustivamente testadas em funcionalidade e desempenho. Apesar disso os resultados apresentados na Tabela 3 fornecem evidências de um aumento de tempo linear em relação à demanda de solicitações de autenticação, indicando um bom desempenho em termos de escalabilidade.

6 CONCLUSÕES

Este trabalho apresentou um modelo e um protótipo de um sistema para gerência e controle de acesso dinâmico para a plataforma OBAA-MILOS.

A utilização de tecnologias como agentes inteligentes de software, em conjunto com a base ontológica definida para o controle de acesso, possibilitou a integração aos agentes e serviços já existentes na plataforma. A adaptação das aplicações da plataforma para utilizarem os mecanismos de autenticação e autorização do sistema *Fayol* é facilitada pela utilização do framework JADE já incorporado ao ambiente.

O funcionamento do sistema, com seus mecanismos semânticos de autenticação e autorização, evidenciados nos experimentos realizados de forma eficaz e compatível às características da plataforma, permitem concluir que os objetivos propostos foram atingidos.

A performance medida no experimento da seção 5.4 demonstrou que supera as necessidades atuais da plataforma e, a possibilidade de escalabilidade facilitada com a criação de vários *containers* JADE que hospedariam um ou vários agentes de controle de acesso, utilizando servidores diferentes no ambiente da plataforma, supririam a necessidade emergencial de crescimento em número de usuários, agentes e recursos do sistema sem um grande impacto na configuração para atender a esta possível demanda.

No decorrer do desenvolvimento deste trabalho verificou-se que os sistemas tradicionais para controle de acesso, quando aplicados a ambientes de computação em nuvem, necessitam de grande esforço para configuração e operacionalização. Adaptações no sentido do tratamento dos diversos agentes e serviços envolvidos nestes ambientes, são necessárias.

Existe a possibilidade de avanço neste trabalho no sentido de atender a gerência e o controle de acesso em ambientes de computação em nuvem. Modelos de autenticação estão consolidados e são utilizados com sucesso em vários ambientes. O ponto a ser estudado em ambientes de computação em nuvem é a autorização. A utilização de técnicas de coleta de dados de localização, de funcionalidades e mesmo limitações de agentes e serviços envolvidos em um ambiente de computação em nuvem poderiam enriquecer a ontologia de acesso, permitindo o processamento de inferências para definir a autorização de requisições, reduzindo e facilitando o trabalho de gerência de configuração do controle de acesso nestes ambientes.

Concluímos desta forma, que o sistema *Fayol*, caracteriza-se como um protótipo de sistema de gerência e controle de acesso baseado em uma ontologia de acesso e agentes de software, posicionando-se em uma área de pesquisa recente, relacionada a ambientes e plataformas onde o tratamento semântico está inerentemente integrado à sua operação. A utilização de uma base semântica permite avançar o estudo de sistemas de controle de acessos dinâmicos e inteligentes, permitindo o processamento de inferências de ontologias de acesso e possíveis benefícios com relação ao gerenciamento das políticas de segurança.

Além disso, nenhum dos trabalhos relacionados que empregam essa nova abordagem semântica (ver Seções 3.1 e 3.2) definem funcionalidades de autenticação e autorização em um

ambiente federado, que é uma característica exclusiva do sistema *Fayol*, no contexto de sistemas de controle de acesso baseados em ontologias. Isso constitui uma inovação do *Fayol*, no contexto dessa nova abordagem ontológica e baseada em agentes para construção de serviços de autenticação e autorização.

REFERÊNCIAS

ABHISHEK, K.; ROSHAN, S.; KUMAR, P.; RANJAN, R. A Comprehensive Study on Multifactor Authentication Schemes. In: ACITY (2), 2012. **Anais...** [S.l.: s.n.], 2012. p. 561–568.

BOOCH, G.; JACOBSON, I.; RUMBAUGH, J. **The Complete Uml Training Course**. [S.l.]: Prentice Hall PTR, 2000.

BRESCIANI, P.; PERINI, A.; GIORGINI, P.; GIUNCHIGLIA, F.; MYLOPOULOS, J. Tropos: an agent-oriented software development methodology. **Autonomous Agents and Multi-Agent Systems**, [S.l.], p. 203–236, 2004.

CASTRO, J.; ALENCAR, F.; SILVA, C. **Engenharia de Software Orientada a Agentes**. [S.l.]: Pontifícia Universidade Católica, Rio de Janeiro, RJ, 2006.

CERVENKA, R. Modeling Multi-Agent Systems with AML. In: SOFTWARE AGENTS, AGENT SYSTEMS AND THEIR APPLICATIONS, 2012. **Anais...** [S.l.: s.n.], 2012. p. 9–27.

CHOI, C.; CHOI, J.; KIM, P. Ontology-based Access Control Model for Security Policy Reasoning in Cloud Computing. **J. Supercomput.**, Hingham, MA, USA, v. 67, n. 3, p. 711–722, Mar. 2014.

FERRAIOLO, D. F.; KUHN, D. R.; SANDHU, R. S. RBAC Standard Rationale: comments on "a critique of the ansi standard on role-based access control". **IEEE Security and Privacy**, [S.l.], p. 51–53, 2007.

FERRAIOLO, D.; KUHN, D.; CHANDRAMOULI, R. **Role-based Access Control**. [S.l.]: Artech House, 2003. (Artech House computer security series).

GANDON, F.; SCHREIBER, G. **RDF 1.1 XML Syntax**. [S.l.]: W3C, 2014. W3C Recommendation, http://www.w3.org/TR/2014/REC-rdf-syntax-grammar-20140225/.

GLUZ, J. C. **Introdução à infraestrutura MILOS**. Pós-Graduação em Computação Aplicada (PIPCA): Universidade do Vale do Rio dos Sinos (UNISINOS), São Leopoldo, RS, 2010. Disponível em: <http://obaa.unisinos.br/>. Acesso em: 18 out. 2013.

GOLEMAN, D. **OS MESTRES DA ADMINISTRAÇAO**: adam smith, c. k. prahalad, daniel goleman. [S.l.]: CAMPUS, 2007.

GÓMEZ-PÉREZ, A.; FERNANDEZ-LOPEZ, M.; CORCHO, O. **Ontological Engineering**: with examples from the areas of knowledge management, e-commerce and the semantic web. first edition. [S.l.]: Springer, 2004. (Advanced Information and Knowledge Processing).

ITU-T. **Recommendation Z.151 (11/08)**: user requirements notation (urn) - language definition. Geneva, Switzerland: [s.n.], 2012.

JENA. **Framework for Building Semantic Web Applications**. [S.l.: s.n.], 2013. Disponível em: <http://jena.apache.org/>. Acesso em: 9 dez. 2013.

KATAL, A.; GUPTA, P.; WAZID, M.; GOUDAR, R.; MITTAL, A.; PANWAR, S.; JOSHI, S. Authentication and authorization: domain specific role based access control using ontology. In: INTELLIGENT SYSTEMS AND CONTROL (ISCO), 2013 7TH INTERNATIONAL CONFERENCE ON, 2013. **Anais...** [S.l.: s.n.], 2013. p. 439–444.

LAMPSON, B. W. Protection. **Operating Systems Review**, [S.l.], p. 18–24, 1974.

MOTIK, B.; PARSIA, B.; PATEL-SCHNEIDER, P. **OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition)**. [S.l.]: W3C, 2012. W3C Recommendation, http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/.

NOVELL. **LDAP Classes for Java**. [S.l.: s.n.], 2013. Disponível em: <http://www.novell.com/developer/ndk/>. Acesso em: 30 jun. 2014.

OpenLDAP. **Open Source Implementation of the Lightweight Directory Access Protocol**. [S.l.: s.n.], 2014. Disponível em: <http://www.openldap.org/>. Acesso em: 28 jun. 2014.

POLLERES, A.; GEARON, P.; PASSANT, A. **SPARQL 1.1 Update**. [S.l.]: W3C, 2013. W3C Recommendation, http://www.w3.org/TR/2013/REC-sparql11-update-20130321/.

RAMACHANDRAN, J. **Designing Security Architecture Solutions**. [S.l.]: Wiley, 2002. (Wiley Desktop Editions Series).

SEABORNE, A.; HARRIS, S. **SPARQL 1.1 Query Language**. [S.l.]: W3C, 2013. W3C Recommendation, http://www.w3.org/TR/2013/REC-sparql11-query-20130321/.

SHADBOLT, N.; BERNERS-LEE, T.; HALL, W. The Semantic Web Revisited. **IEEE Intelligent Systems**, [S.l.], p. 96–101, 2006.

SILVA, L. R. J. da; GLUZ, J. C. **MSSearch**: busca semântica de objetos de aprendizagem obaa com suporte a alinhamento automático de ontologias. Universidade do Vale do Rio dos Sinos (UNISINOS) - São Leopoldo/RS: Programa Interdisciplinar de Pós Graduação em Computação Aplicada (PIPCA), 2012.

Spring Security. **Framework for Authentication and Authorization to Java Applications**. [S.l.: s.n.], 2014. Disponível em: <http://projects.spring.io/spring-security/>. Acesso em: 28 jun. 2014.

SUNGAILA, M. **Autenticação Centralizada com OpenLDAP**: integrando serviços de forma simples e rápida. [S.l.]: NOVATEC, 2007.

VICARI, R. M.; BEZ, M.; SILVA, J. M. C. da; RIBEIRO, A.; GLUZ, J. C.; SANTOS, E.; PRIMO, T.; BORDIGNON, A. **Proposta de Padrão de Objetos de Aprendizagem Baseados em Agentes (OBAA)**. [S.l.]: UFRGS/CINTED/UNISINOS, Porto Alegre e São Leopoldo, RS, 2010. Disponível em: <http://www.portalobaa.org/padrao-obaa/artigos-publicados/>. Acesso em: 24 out. 2013.

VICARI, R. M.; GLUZ, J. C. **Infraestrutura OBAA-MILOS**: infraestrutura multiagente para suporte a objetos de aprendizagem OBAA. [S.l.]: UFRGS/CINTED, Porto Alegre, RS, 2011.

WOOLDRIDGE, M. **An Introduction to MultiAgent Systems**. [S.l.]: Wiley, 2008.

YU, E. S. K. Social Modeling and i*. In: CONCEPTUAL MODELING: FOUNDATIONS AND APPLICATIONS, 2009. **Anais...** [S.l.: s.n.], 2009. p. 99–121.