UNISINOS

Programa Interdisciplinar de Pós-Graduação em

Computação Aplicada

Mestrado Acadêmico

Ederson Luiz Silveira

Sistema LINNAEUS: Apoio Inteligente para a Catalogação e Edição de Metadados de Objetos de Aprendizagem

São Leopoldo, 2013

UNIVERSIDADE DO VALE DO RIO DOS SINOS - UNISINOS
UNIDADE ACADÊMICA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA INTERDISCIPLINAR DE PÓS-GRADUAÇÃO
EM COMPUTAÇÃO APLICADA
NÍVEL MESTRADO

EDERSON LUIZ SILVEIRA

Sistema LINNAEUS: Apoio Inteligente para a Catalogação e Edição de Metadados de Objetos de Aprendizagem

São Leopoldo
2013

Ederson Luiz Silveira

Sistema LINNAEUS: Apoio Inteligente para a Catalogação e Edição de Metadados de Objetos de Aprendizagem

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

Orientador: Dr. João Carlos Gluz

São Leopoldo

2013

S587s Silveira, Ederson Luiz

Sistema LINNAEUS : apoio inteligente para a catalogação e edição de metadados de objetos de aprendizagem / por Ederson Luiz Silveira. – 2013.

113 f. : il., 30cm.

Dissertação (mestrado) — Universidade do Vale do Rio dos Sinos, Programa de Pós-Graduação em Computação Aplicada, 2013.
Orientador: Prof. Dr. João Carlos Gluz.

1. Objetos de aprendizagem. 2. Metadados. 3. Agentes inteligentes. I. Título.

CDU 004.89

004:37

Catalogação na Fonte:

Bibliotecária Vanessa Borges Nunes - CRB 10/1556

Ederson Luiz Silveira

Sistema LINNAEUS: Apoio Inteligente para a Catalogação e Edição de Metadados de Objetos de Aprendizagem

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)

_____

_____

Dedico esta dissertação a memória de meu pai, Agenor Silveira, pessoa dedicada e preocupada em fazer o bem a outras pessoas sem ter a intenção de receber algo em troca.

AGRADECIMENTOS

Agradeço primeiramente a Deus, motivo maior de todas as coisas, que me acompanha e me guia dia-a-dia em minha jornada e me dá forças para lutar para seguir em frente.

A realização deste trabalho não seria possível sem a ajuda de algumas pessoas e organizações que eu quero expressar minha gratidão nesta dissertação. Ao meu orientador Profº Dr. João Carlos Gluz que me auxiliou durante todo o caminho até a conclusão desta dissertação, e me ajudou na compreensão e no esclarecimento de minhas dúvidas. Ao projeto OBAA-MILOS pela bolsa, que subsidiou esta dissertação e forneceu a oportunidade de desenvolver um bom trabalho e alcançar o objetivo que venho buscando, de obter o título de mestre.

A minha esposa, amiga e companheira de todas as horas e momentos, Graziela da Silva Genari por estar sempre ao meu lado e me apoiar neste período de muito trabalho e estudos.

A minha mãe Maria Lourdes Silveira e aos meus irmãos Mauro Silveira e Clodoaldo Silveira por serem a base forte da minha vida e entenderem a minha ausência do meio familiar.

A Marcio Orestes Hack pela compreensão da minha ausência, o auxilio e socorro a meus colegas de trabalho durante este período de faltas.

Ao meu amigo e colega de projeto Matheus Campezatto, que sem a sua ajuda e dedicação no desenvolvimento da ferramenta, proporcionou a oportunidade para que este trabalho pudesse ser concluído.

Por fim agradeço a todos que de alguma maneira me ajudaram neste período direta ou indiretamente.

RESUMO

A tecnologia de objetos de aprendizagem (OA) vêm tendo uma crescente utilização em contexto educacional, oferendo recursos digitais que estão cada vez mais presentes em salas de aula e na educação a distância. Em muitos casos, esta tecnologia está presente nos ambientes educacionais de forma implícita, sem que seja necessário saber que os materiais educacionais estão incorporados em objetos de aprendizagem. Como sua utilização cresce a cada ano, surge a necessidade de ferramentas que auxiliem o processo de catalogação e de edição dos metadados destes objetos, para tornar possível a sua recuperação, reutilização e alteração dos objetos. Para prover apoio a este processo de catalogação, a presente dissertação apresenta o sistema *Linnaeus* de suporte a catalogação de objetos de aprendizagem e edição de metadados, que através da junção das tecnologias de agentes inteligentes e da web semântica, na forma de *wizards* e de ontologias educacionais, se propõe a fornecer um apoio inteligente e pró-ativo, ajudando usuários sem conhecimentos técnicos sobre metadados ou padrões de objetos de aprendizagem a catalogar de forma correta seus objetos.

Palavras-Chave: Objetos de Aprendizagem. Metadados. Agentes Inteligentes.

ABSTRACT

Learning objects (LO) are getting more importance in the education environment, providing digital resources that are increasingly common in both classrooms and e-learning teaching. Nowadays, these objects can be found in educational materials, without being explicitly defined as learning objects. During the latest years the use of learning objects is growing up, bringing with that the necessity of solutions for the cataloging and editing metadata information about these objects to make possible the recovery, reuse and even the change of these objects. To give support for this task, this dissertation proposes an intelligent learning object metadata editor, the Linnaeus system, which will provide help for the cataloging process of learning objects through the use of intelligent agents, and semantic web technologies, in the form of editing wizards and learning domain educational ontologies.

Keywords: Learning Objects, Metadata. Intelligent agent.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ADL	Advanced Distributed Learning Initiative
API	Application Programming Interface
API SDB	Application Programming Interfaces Suitable DataBase
AUML	Agent Unified Modeling Language
BIOE	Banco Internacional de Objetos Educacionais
BPM	Business Process Management
CLIPS	C Language Integrated Production System
CSELT	Centro Studi E Laboratori Telecomunicazioni
DCMI	Dublin Core Metadata Initiative
DL	Description Logic
EaD	Ensino a Distância
FIPA	Foundation for Intelligent Physical Agents
FIPA ACL	FIPA – Agent Communication Language
FIPA-OS	FIPA – Operating System
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IA	Inteligência Artificial
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IEEE-LOM	IEEE – Learning Object Metadata
IEEE-LTSC	IEEE – Learning Technology Standards Committee
IMS	Inovation Adoption Learning
ISO	International Services for Standartization
JADE	Java Agent DEvelopment framework
JAVA	Linguagem de Programação Orientada a Objeto
JENA	Freamwork para Web Semântica
JENA TDB	Freamwork para dados semântico RDF
JESS	Java Expert System Shell
JSF	Java – Server Faces
JSR-94	Java Specification Request 94
KQML	Knowledge Query Manipulation Language
LOM	Learning Object Metadata
MILOS	Multagent Infraestrutcture for Learning Object Support
MPEG	Moving Picture Experts Group
NASA	National Aeronautics and Space Administration
OA	Objeto de Aprendizagem
OBAA	Objetos de Aprendizagem Suportados por Agentes
OWL	WEB Ontology Language
OWL-DL	Web Ontology Language – Description Logic
Pellet	Java open-source OWL DL reasoner
RDF	Resource Definition Format

RDQL	RDF Data Query Language
RETE	Algoritmo Utilizado em Mecanismos de Inferência
RIF	Rule Interchange Format
SBTVD	Sistema Brasileiro de Televisão Digital
SCORM	Sharable Content Object Reference Model
SKOS	Simple Knowledge Organization System
SPARQL	SPARQL Protocol and RDF Query Language
SPI	Service Programming Interface
SROIQ	Extensão de OWL-DL
SWRL	Semantic Web Rule Language
TDB	Truples DataBase
TV	Televisão
UML	Unified Modeling Language
URI	Uniform Resource Identifiers
W3C	World Wide Web Consortium
WADE	Workflows and Agents Development Environment
Web	System of Interlinked Hypertext Documents
WOLF	WOrkflow LiFe cycle management environment
XML	Extensible Markup Language
XML-S	Extensão XML para regras de inferência

SUMÁRIO

1 INTRODUÇÃO

Com o crescente emprego da internet em todos os âmbitos do cotidiano, além da grande variedade existente de ferramentas direcionadas a educação, aumenta a possibilidade de tornar a mediação digital entre professor e aluno mais didática e produtiva. Um Objeto de Aprendizagem (OA) é mais um recurso didático digital para auxiliar professores e alunos.

Entretanto, para que um OA possa ser localizado, utilizado e eventualmente alterado pelo professor, este deve ter suas informações de catalogação devidamente preenchidas e registradas de acordo com o domínio de ensino, o tipo de metodologia educacional, a sua localização na web, a plataforma digital, dentre outras informações a respeito do objeto.

No contexto da padronização os OA são vistos como artefatos descritos em duas camadas (ou níveis):

- Camada dos Metadados: que engloba as informações de catálogo do objeto de aprendizagem, informando sua localização na base de dados, domínio do objeto, aplicação, plataforma de operação, etc... Dentre as informações com significativa importância na aplicação educacional destacam-se dados descritivos utilizados em busca, localização, recuperação e apresentação do conteúdo;
- Camada de Conteúdo: que contém o material de aprendizado que deve ser visualizado pelo usuário para atingir os objetivos de determinado tópico de ensino.

Assim, o processo de catalogação é operacionalizado através do preenchimento de metadados padronizados. Este processo mais limitado de catalogação de OA, baseado em padrões de metadados, opõe-se a grande quantidade de informações que pode ser obtida em buscas na *web*, devido as informações da web estarem armazenadas sem padrões públicos e universais de organização, ou classificação, tendo apenas o objetivo de tornar disponível material para leitura e interpretação por pessoas. O armazenamento destas informações de forma desestruturada torna a tarefa de busca e recuperação de informações da *web* ineficiente em diversos casos.

A catalogação de metadados de um OA deve ser realizada com o auxílio de ferramentas que forneçam ajuda ao usuário na tarefa de criação correta dos metadados, atendendo as necessidades de estruturação e organização correta destas informações, para sua utilização futura não somente por pessoas, mas também por mecanismos de busca.

A elaboração dos metadados pode em alguns momentos gerar um trabalho desnecessário e desgastante para o criador do OA. Isto ocorre porque, para que sejam preenchidos de forma correta, muitas vezes os metadados requerem conhecimentos técnicos desconhecidos pelo usuário, gerando por vezes a desistência da catalogação e a criação de OA com metadados parcialmente preenchidos ou até sem metadados.

A utilização de ferramentas para autoria de OA pode auxiliar no processo de catalogação desde que esta ferramenta de o suporte necessário ao usuário no processo de preenchimento dos metadados. Em alguns casos as informações solicitados pela ferramenta de catalogação são de nível de meta-metadados não levando em consideração, por exemplo, a aplicação do objeto, sua plataforma de trabalho, áreas de conhecimento, etc... gerando um catalogo incompleto de metadados.

Esta proposta de dissertação está voltada para as pesquisas sobre sistemas de catalogação de OA, que serão considerados ferramentas de software que dão apoio à autoria dos metadados de um objeto de aprendizagem, ou seja, no contexto dessa dissertação catalogar um OA equivale a criar e editar seus metadados.

Em termos de padrão de metadados, serão utilizados os metadados especificados pela proposta de padrão de metadados OBAA (Vicari e tal. 2010; Bez et. al 2010). Essa escolha se justifica pela amplitude e generalidade do padrão OBAA. Este padrão para metadados de Objetos de Aprendizagem é definido com base no padrão de metadados IEEE-LOM (IEEE-LTSC, 2002), estendendo o IEEE-LOM pela inclusão de metadados de suporte para a adaptabilidade e interoperabilidade de OA em diversas plataformas de operação como *web*, TV Digital e dispositivos móveis, além de metadados de suporte a acessibilidade por pessoas portadoras de necessidades especiais. O padrão OBAA busca a independência juntamente com flexibilidade, não necessitando de tecnologias proprietárias, e permitindo que inovações tecnológicas sejam acrescidas ao padrão, sem perder a compatibilidade com o material já desenvolvido (Bez et al., 2010).

Outro objetivo do presente trabalho é garantir a integração do sistema de catalogação desenvolvido durante as pesquisas da dissertação (preliminarmente denominado de sistema *LINNAEUS*) à infraestrutura MILOS (GLUZ, VICARI e PASSERINO, 2012), fornecendo parte do serviço de autoria de OA a ser disponibilizado pela infraestrutura. Assim o *LINNAEUS* proverá serviços através do intercâmbio de informações com os demais subsistemas da infraestrutura.

Quanto à estrutura desta dissertação, o documento está organizado como se segue; primeiramente esta introdução, e logo após serão abordados temas relacionados aos fundamentos deste trabalho, incluindo: o capítulo 2 sobre apresentação, metodologia e objetivos. No capítulo 3 sobre a fundamentação teórica pesquisada e utilizada neste trabalho. O capítulo 4 descreve os trabalhos relacionados utilizados como a base comparativa do trabalho. O capítulo 5 apresenta um estudo de caso para ilustrar o uso do protótipo, incluindo a arquitetura do sistema, seus agentes e comunicação entre eles, e também as regras lógicas desenvolvidas para o mecanismo de inferência. Enquanto o capítulo 6 apresenta a validação e resultados de testes executados com o sistema.

Por fim, o Capítulo 7 traz as considerações finais obtidas com o desenvolvimento da pesquisa e também a prospecção de trabalhos futuros.

2 QUESTÃO DE PESQUISA

Este capítulo apresenta uma visão geral sobre as questões que estimularam a presente pesquisa, os objetivos que deverão ser alcançados por ela, juntamente com a metodologia a ser utilizada na solução do problema.

2.1 Apresentação

Em um contexto geral esta pesquisa pretende abordar o problema da catalogação dos metadados de Objetos de Aprendizagem. Utilizando neste processo técnicas, como a inferência sobre o conteúdo de ontologias para criação de novos metadados, aplicadas em uma ferramenta de catalogação que apoiará professores, profissionais de áreas pedagógicas, a montar catálogos de objetos compatíveis com a proposta de padrão de objetos de aprendizagem OBAA (Vicari et al. 2010).

O problema está relacionado com a complexidade e extensão dos padrões atuais de metadados de OA, incluindo IEEE-LOM (IEEE-LTSC, 2002), Dublin-Core (DCMI, 2008) e OBAA (Vicari et al. 2010; Bez et. al 2010), e ao desconhecimento por parte de professores e *designers* de materiais educacionais em relação às características e necessidades técnicas para a autoria destes metadados, provocando muitas vezes a desistência da conclusão e o correto preenchimento dos metadados do OA.

Assim, o problema de pesquisa pode ser resumido em:

Verificar a viabilidade de um sistema de catalogação, que utilize as tecnologias combinadas de agentes inteligentes e web semântica, incluindo suporte de agentes wizards e de ontologias, pode contribuir para que se obtenha uma catalogação prática e correta de Objetos de Aprendizagem.

2.2 Objetivos

A questão de pesquisa dessa dissertação implica nos objetivos gerais de projetar, desenvolver e testar um sistema computacional, baseado em agentes inteligentes e na web semântica que possa oferecer um apoio efetivo para o processo de catalogação de objetos de aprendizagem, suportados pela proposta OBAA-MILOS.

Assim os objetivos gerais desta proposta podem ser resumidos em

- Analisar as soluções atuais de catalogação de OA a fim de especificar um processo de catalogação baseado em ontologias educacionais e no uso de *wizards* de catalogação capazes de interpretar os conhecimentos dessas ontologias para ajudar no processo de catalogação
- Projetar um sistema para catalogação formado por agentes especialistas na elaboração, criação e alteração de metadados e por agentes *wizards* capazes de aplicar os conhecimentos adquiridos de ontologias de domínios educacionais no processo de catalogação inteligente;

- Desenvolver um protótipo desse sistema que possa operar em um ambiente *web*;
- Planejar e conduzir experimentos e testes de forma que os métodos utilizados na catalogação possam ser avaliados e validados;

Esses objetivos gerais serão atingidos através das seguintes atividades de pesquisa:

- Projetar, desenvolver e testar uma interface para o usuário, baseada em sistemas multiagentes, que seja apropriada para a catalogação inteligente de metadados;
- Especificar as propriedades das ontologias educacionais que serão empregadas no processo de catalogação;
- Especificar detalhadamente as propriedades dos agentes *wizards* que serão responsáveis pelos processos de auxilio a professores e desenvolvedores de OA no processo de elaboração dos metadados;
- Modelar o comportamento destes agentes e desenvolver um protótipo do mesmo;
- Especificar e desenvolver a comunicação entre os agentes e as facilidades de interface do sistema MILOS;
- Planejar e conduzir testes empíricos em laboratório e também através de acesso remoto;
- Empreender uma análise crítica destes resultados.

2.3 Metodologia

A metodologia que será utilizada neste projeto segue as diretrizes técnicas e metodológicas adotadas na infra-estrutura MILOS (GLUZ, VICARI e PASSERINO, 2012) que prevê o uso de camadas de ontologias, sistemas de agentes inteligentes e facilidades de interface.

O sistema multiagente de catalogação de metadados a ser projetado e desenvolvido no decorrer da dissertação será denominado de *Linnaeus*. Os protótipos servirão para explorar os limites da tecnologia, além de esclarecer dúvidas e identificar necessidades e possibilidades de um sistema inteligente de catalogação de OA.

O primeiro protótipo a ser desenvolvido do Sistema *Linnaeus* tem como objetivo definir e testar as características da interface principal do *Linnaeus*, buscando identificar requisitos de usabilidade para sistemas de catalogação. Também serão exploradas as formas de integração dos agentes *wizards* com a interface principal e os meios de obter as informações dos usuários que são necessárias para a seleção de *wizards*.

Posteriormente serão desenvolvidos protótipos para os agentes *wizards* com base em ontologias educacionais previamente desenvolvidas para a MILOS, além dos agentes de gerenciamento do sistema e da comunicação entre esses agentes.

A previsão é que os protótipos sejam desenvolvidos na linguagem Java orientada a objetos, com suporte de plataforma JADE para comunicação entre agentes. Esta linguagem e plataforma foram escolhidas por sua versatilidade, eficiência, portabilidade e segurança.

3 FUNDAMENTAÇÃO DA PESQUISA

Este capítulo apresenta as bases tecnológicas do Sistema *Linnaeus*. Este sistema depende principalmente das tecnologias relacionadas aos objetos de aprendizagem, incluindo a proposta de metadados OBAA (VICARI et al, 2010). Também são importantes para o projeto as tecnologias de agentes inteligentes, sistemas multiagente e a infra-estrutura de agentes MILOS (GLUZ, VICARI e PASSERINO, 2012

3.1 Agentes

Um agente inteligente pode ser descrito como “um sistema autônomo que apresenta um comportamento que é determinado por um processo de raciocínio baseado na representação de suas atitudes, tais como crenças, comprometimentos e desejos” (WOOLDRIDGE; JENNINGS, 1994).

De acordo com Lind (2006) “a partir de observações feitas no ambiente onde está inserido um agente inteligente constrói sua base de conhecimento acerca dos objetivos para os quais foi desenvolvido”. A forma utilizada pelos agentes para adquirir conhecimento no ambiente é através da realização de inferências lógicas sobre as observações (percepções) de seu interesse. Um agente utiliza a sua base de conhecimento para alcançar seus objetivos, ou seja, o agente realiza inferências sobre suas crenças para conhecer se determinada ação será executada.

Jennings (1995) e Franklin (1996) evidenciam algumas características importantes para um agente de acordo com o contexto do agente e seu papel no sistema:

- Autonomia: um agente é capaz de agir sem intervenção humana direta de acordo com seu próprio controle e estada interna.
- Habilidade Social: um agente é capaz de interagir com outro agente, auxiliando ou pedindo auxílio para atingir seus objetivos.
- Reatividade: um agente pode perceber e reagir a qualquer alteração no ambiente.
- Proatividade: um agente pode agir por conta própria, sem a necessidade de estímulos externos.

3.1.1 Engenharia de *Software* Orientada a Agentes

Lind (2006) compara as arquiteturas tradicionais de *software* com a Engenharia de *Software* Orientada a Agentes: as arquiteturas de *software* que possuem muitos componentes que interagem entre si, cada um com seu próprio objetivo, e se relacionam em protocolos complicados, são, tipicamente mais difíceis de ter seu projeto correto e eficiente do que programas que não se relacionam com outros. Agentes inteligentes se enquadram bem neste complexo cenário provendo uma forma mais intuitiva e nativa de interação entre sistemas.

Diversas aplicações do mundo real se enquadram no cenário mencionado: a computação ubíqua, redes de sensores, computação em rede, etc. Como consequência o desenvolvimento de sistemas orientados a agentes tem se tornado uma prática cada vez mais presente e requisitada no desenvolvimento de *software*.

A partir dessa situação, o papel da engenharia de *software* é prover estruturas e técnicas que tornam mais fácil trabalhar com tal complexidade (JENNINGS, 2002). Neste contexto, nos últimos anos os pesquisadores têm considerado novas abordagens, tal como a utilização de agentes, a fim de melhorar significantemente o processo de desenvolvimento de software complexo (CASTRO, J; ALENCAR, F e SILVA, C; 2006).

As técnicas orientadas a agentes são adequadas para o desenvolvimento de sistemas complexos de *software* porque (JENNINGS e WOOLDRIDGE 2001):

- As decomposições orientadas a agentes são uma maneira efetiva de dividir o espaço do problema de um sistema composto por várias partes.
- Ao mesmo tempo as abstrações chave presentes no modo de pensar orientado a agentes são um meio natural de modelar este tipo de sistema.
- A programação orientada a agentes também oferece um meio apropriado de desenvolver e gerenciar as dependências e interações que existem em um sistema complexo.

Atualmente existe uma série de tecnologias que fornecem suporte ao desenvolvimento de sistemas orientados a agentes. Através do uso destas tecnologias é possível desenvolver projetos de *software* utilizando o paradigma de desenvolvimento de *software* orientado a agentes.

Para fornecer apoio ao projeto e desenvolvimento de um sistema complexo é indispensável à utilização de uma metodologia baseada em técnicas de engenharia de *software*. Uma metodologia de projeto e desenvolvimento de *software* é tipicamente caracterizada por dois aspectos (BAUER; ODELL; UML, 2002):

- Linguagem de modelagem: utilizada para descrever modelos, definições de componentes e seu comportamento, definições de uma notação padrão e seu significado;
- Processo de *software*: define as atividades de desenvolvimento, e suas formas de relacionamento.

A engenharia de *software* orientada a agentes considera desde o início as possibilidades de se modelar o *software* do ponto de vista dos agentes. Neste nível de abstração os agentes são entidades atômicas que se comunicam para desenvolver funcionalidades do sistema. A comunicação entre essas entidade deve ser suportada por uma linguagem de comunicação entre agentes, como FIPA ACL, e por uma ontologia utilizada para associar um significado às mensagens (BERGENTI; POGGI, 2000).

O uso de linguagens padronizadas para modelagem de sistemas multiagentes tem sido pesquisado pela comunidade de engenharia de *software* orientada a agentes. Como resultado, alguns trabalhos consideram a UML (Unified Modeling Language) (FOWLER, 2000) como uma solução para este problema. Baseado no sucesso da UML, padrão de fato para modelagem de sistemas orientados a objetos, foi desenvolvida a linguagem AUML (Agent UML) (BAUER; MULLER; ODELL, 2001) para facilitar o processo de modelagem de agentes. A AUML é uma extensão da linguagem UML que tenta padronizar a forma de projetar sistemas multiagentes. Até o surgimento da AUML as demais linguagens de modelagem estavam altamente ligadas a um processo de *software* ou a alguma tecnologia, ferramenta ou linguagem de programação, que algumas vezes podia ser comercial. Além disso, a utilização de UML traz outros benefícios:

- Disponibilidade de ferramentas e notação UML para acompanhar todo ciclo de desenvolvimento de *software*;
- Crescimento da UML como padrão de fato na comunidade de engenharia de *software*;
- Grande quantidade de recursos humanos e material para UML;
- Linguagem em evolução constante.

3.1.2 Sistemas Multiagentes e Linguagem de Comunicação entre Agentes

Os sistemas multiagentes são compostos por múltiplos agentes inteligentes de *software*. Estes componentes podem estar em uma mesma estação de trabalho ou espalhados em uma rede. Sistemas com estas características, normalmente são utilizados para resolver problemas considerados difíceis ou impossíveis de serem resolvidos por um único agente. Estes agentes, quando inseridos em um ambiente como uma comunidade, interagem com os demais para formar suas crenças a partir do conhecimento já adquirido pelos demais agentes componentes do sistema (WOOLDRIDGE, 2002).

O sistema multiagente pode ser utilizado para resolver problemas com múltiplas perspectivas. Esses sistemas têm a vantagem de possibilitar a resolução de problemas tradicionais de forma distribuída e concorrente. Além disto, através de uma arquitetura com mais de um agente de *software* é possível obter padrões de interações mais sofisticados. Exemplos de tipos comuns de interação: cooperação (trabalhar em conjunto por um objetivo comum); problema de coordenação (organização da atividade para solução do problema de modo que interações prejudiciais sejam evitadas), e de negociação (atingir a um acordo aceitável para todas as partes envolvidas).

Uma arquitetura de sistemas baseada na comunicação entre diversos agentes de *software* garante ao sistema a ótima capacidade de integração. De acordo com Jennings, Sycara e Wooldridge (1998), através da "flexibilidade e da natureza de alto nível das interações inerentes dos sistemas multiagentes pode-se distinguir positivamente esta arquitetura de *software* de outras formas de arquitetura".

Em um ambiente com múltiplos agentes a comunicação entre estes elementos de *software* é um ponto crucial. Para garantir baixa coesão e alto acoplamento as interfaces e os protocolos devem seguir padrões que garantam que os componentes do sistema possam interagir de forma transparente independente de linguagem de desenvolvimento, tecnologia ou plataforma nas quais foram concebidos. Para atender estes requisitos, em 2005, a IEEE propôs a criação de uma entidade que organizasse e determinasse padrões para interoperabilidade entre agentes, a *Foundation of Intelligent Physical Agents* (FIPA) (2000).

A FIPA utilizou como base padrões anteriores, o *Knowledge Query Manipulation Language* (KQML), que com este conhecimento determinou uma Linguagem de comunicação entre agentes composta por 21 atos comunicativos. Estes atos comunicativos são mensagens em um formado predeterminado em que cada uma destas mensagens representa algum objetivo de comunicação.

Tabela 1 – Atos comunicativos da FIPA

Ato Comunicativo	Passar Info.	Requisitar Info.	Negociar	Executar Ações	Atender Erros
accept-proposal			X		
agree				X	

cancel		X		X	
Cfp			X		
confirm	X				
disconfirm	X				
failure					X
inform	X				
inform-if	X				
inform-ref	X				
not-understood					X
propose			X		
query-if		X			
query-ref		X			
refuse				X	
reject-proposal			X		
request				X	
request-when				X	
request-whenever				X	
subscribe		X			

Fonte: FIPA. (2000).

A FIPA realizou a padronização de algumas interações realizadas entre os agentes através de protocolos. Estes protocolos tomam como base os atos comunicativos que compõem a FIPA-*Agent Communication Language* (FIPA-ACL) para atingir seus objetivos. A utilização de protocolos de interação padrão favorece a transparência de relacionamento nos sistemas multiagentes ajudando na alta coesão e o baixo acoplamento.

Hoje existe uma grande variedade de implementação do padrão FIPA disponível no mercado através de plataformas de comunicação entre agentes compatíveis com as especificações FIPA, como por exemplo, JADE (BELLIFEMINE F.; G. CAIRE, 2002), ZEUS (COLLIS J.; NDUMU, 2007) e FIPA-OS (DEFINE DESIGN, 2007), todas estas plataformas são baseadas na linguagem JAVA e licenciadas como *software* livre.

3.2 JADE

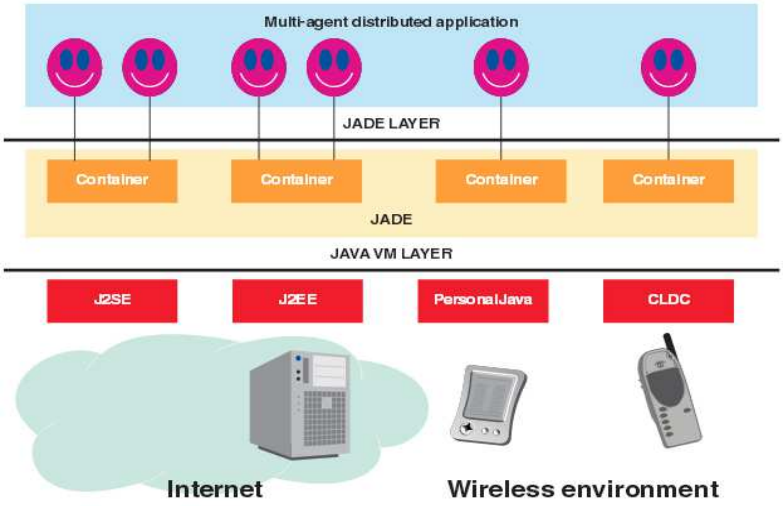
A plataforma de agentes Jade (*Java Agent DEvelopment framework*) (BELLIFEMINE, POGGI et al. 1999) é constituído de um ambiente para desenvolvimento de aplicações baseada em agentes seguindo as especificações da FIPA para interoperabilidade entre sistemas multiagente e com sua implementação feita em Java (BELLIFEMINE, POGGI et al. 2003). Foi desenvolvido e, é mantido pelo CSELT da Universidade de Parma na Itália, sob licença *open source*.

O objetivo da plataforma Jade é simplificar e facilitar o desenvolvimento e a criação de sistemas multiagentes garantindo um padrão de interoperabilidade entre os mesmos através de uma grande quantidade de serviços (BELLIFEMINE, POGGI et al. 2003) que facilitam e possibilitam a comunicação entre agentes, de acordo com as especificações da FIPA. Toda a

comunicação entre agentes é feita através da troca de mensagens FIPA. Além disso, esta plataforma fornece suporte aos aspectos que não fazem parte do agente em si e que são independentes das aplicações. Os serviços disponíveis pelo Jade são: serviço de nomes (*naming service*), páginas amarelas (*yellow-page service*), transporte de mensagens, serviços de codificação e descodificação de mensagens e uma biblioteca de protocolos para interação entre sistemas pronta para utilização.

A plataforma Jade pode ser considerado como um *middleware* de ferramentas para o desenvolvimento de sistemas multiagente. Os agentes podem estar distribuídos em diversos ambientes e estarem conectados entre si por um container Jade em que estes agentes comunicam-se entre si, e podem ser acessados por dispositivos externos através deste mesmo container Jade onde estão contidas as camadas de comunicação (BELLIFEMINE, POGGI et al. 2003).

Figura 1 - Plataforma JADE



Fonte: Bellifemine, Caire et al. (2003).

3.3 WADE

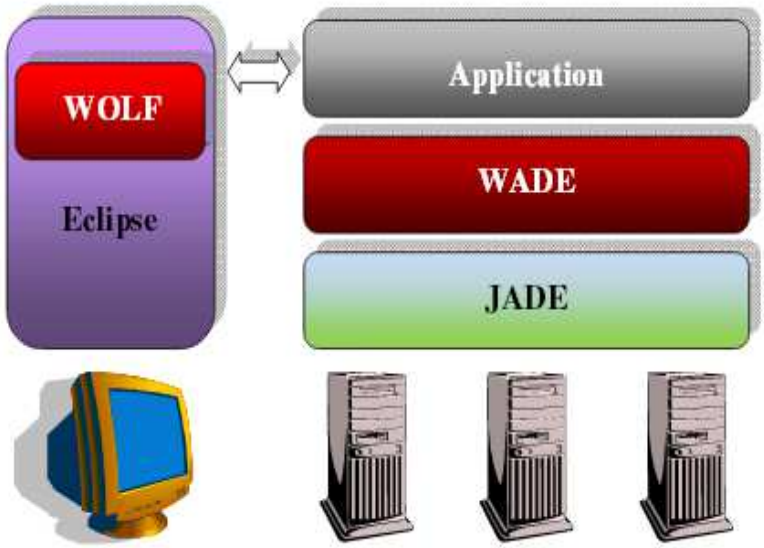
WADE (*Workflows and Agents Development Environment*) é uma plataforma de *software* desenvolvida para facilitar a criação de projetos de sistemas multiagentes distribuídos, em que as tarefas dos agentes podem ser definidas seguindo um fluxograma de operações. O WADE pode ser visto como um *Engine Workflow* de processos de negócio orientado a agente (CAIRE, 2011).

WADE é um subsistema da plataforma JADE, baseado em *workflow* de regras de negócio BPM *(Business Process Management).* Uma das principais vantagens da utilização do fluxo de trabalho é possibilidade de representar comportamentos de sistemas multiagentes composto por diversas partes de forma que sua compreensão seja fácil. Também adicionar a plataforma JADE o apoio a execução de tarefas definidas de acordo com fluxo de trabalho (BPM), com o auxílio de mecanismos para ajudar na administração de processos distribuídos de difícil compreensão (CAIRE, et al, 2008). O WADE tem um ambiente de desenvolvimento chamado WOLF (*WOrkflow LiFe cycle management environment)*, que é um *plugin* para a

IDE (*Integrated Development Environment*) ECLIPSE. Este *plugin* fornece o ambiente de desenvolvimento para agentes baseado em fluxo de trabalho, como mostrado na figura 2.

Na figura 2, o fluxo de trabalho é criado no ambiente WOLF. O Java gera a codificação dos arquivos binários para a execução, o ambiente WADE interpreta o fluxo de trabalho gerado pelo WOLF e o ambiente JADE irá executar o agente em um ambiente distribuído ou local, em conjunto com fluxo de trabalho controlado pelo WADE.

Figura 2 – Plataforma WADE

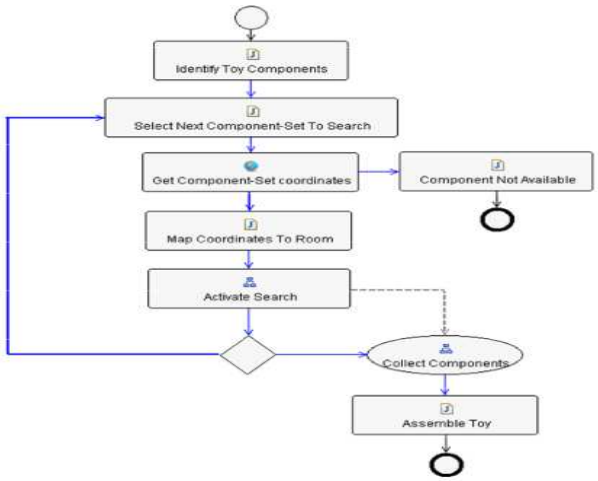


Fonte: Caire. (2011).

O fluxo de trabalho utilizado no WADE, como mencionado anteriormente tem o objetivo de tornar amigável o desenvolvimento da programação orientada a agentes em sistemas distribuídos (BERGANTI, et al. 2012). O fluxo de trabalho BPM torna a visualização e entendimento do sistema mais fácil, por ser um fluxo de trabalho em que a automação dos negócios de uma empresa possui informações ou tarefas que são passadas de um ator para outro de acordo com um conjunto de regras processuais (BERGANTI, et al. 2012).

Estes fluxos de trabalho BPM são muitas vezes utilizados para orientar e conduzir o trabalho de pessoas. A utilização de fluxos BPM baseado a agentes é realizada através de um conjunto de módulos de *software* que atendam aos critérios de granulação grossa, que definem que o agente e seu vizinho estão envolvidos na gestão do trabalho ao longo do processo de negócio, ou seja, é realizada a descentralização de tarefas e estas são distribuídas entre os agentes (BERGANTI, et al. 2012). Na figura 3 é mostrado um exemplo de um fluxo de trabalho de para um agente desenvolvido para o ambiente WADE.

Figura 3 – Exemplo de Fluxograma WADE

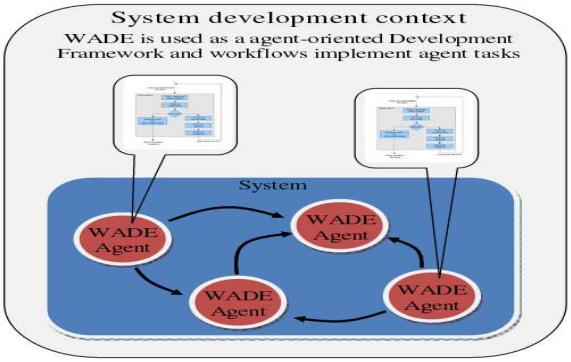


Fonte: Sacchi et al. (2011).

O papel dos agentes WADE, na utilização em sistemas multiagentes não é apenas sobre a exploração da autonomia destes agentes para gestão de situações dinâmicas e imprevisíveis, mas sim trata-se de fornecer aos desenvolvedores ferramentas amigáveis para o auxílio na representação de sistemas multiagentes distribuídos (BERGANTI, et al. 2012). Além disso, a forte integração do WADE com as tecnologias tradicionais de desenvolvimento, como Java e serviços da Web, permite que os desenvolvedores gradativamente possam aceitar as tecnologias de agentes em seus sistemas (BERGANTI, et al. 2012).

A criação de sistemas multiagentes com o auxílio do WADE para descrever o comportamento dos agentes do sistema, requer um conhecimento a nível *expert* em sistemas de agentes em JADE e na linguagem de programação Java. Apesar de possuir interface e fluxo de trabalho amigáveis para a descrição do comportamento de sistemas multiagentes, a utilização e codificação do sistema requer um maior cuidado por parte do desenvolvedor. Porém, após a familiarização com a ferramenta e seu ambiente o desenvolvimento do sistema e dos comportamentos dos agentes deste sistema decorre naturalmente sem maiores problemas. Na figura 4 é mostrado um sistema multiagentes com o comportamento de seus agentes descrito em WADE, demonstrando que os agentes operam normalmente como os agentes que utilizam somente a plataforma JADE.

Figura 4 - Sistema de Agentes Orientado a Fluxo de Trabalho



Fonte: Caire. (2011).

3.4 Ferramentas de Auxílio para Web Semântica

De acordo com a W3C a Web Semântica é a evolução do conceito "*Web* de documentos". A *Web* de Documentos é fundamentada em um conjunto de padrões, que incluem: um mecanismo de identificação global e único URI (*Uniform Resource Identifiers*); um mecanismo de acesso comum o HTTP (*HyperText Transfer Protocol*) e o seu formato padrão para a representação do conteúdo o HTML (*HyperText Markup Language*) (W3C, 2012). Como na *Web* de Documentos a Web de Dados possui alguns padrões base, como o mesmo padrão de identificação e acesso global usado pela *Web* de documentos (as URIs e o HTTP); possui um modelo padrão para representação de dados chamado de RDF(*Resource Description Framework*) e utiliza uma linguagem para realizar consultas na base de dados, a linguagem SPARQL (*SPARQL Protocol and RDF Query Language*) (BIZER, HEATH e BERNERS-LEE, 2009).

A *Web* de dados possibilita que computadores realizem processamento de dados mais útil fornecendo suporte a interações na rede para o desenvolvimento de sistemas. *Web* Semântica é a terminologia do W3C para a *web* de dados ligados (*linked data web*). A *Web* Semântica propicia aos usuários a capacidade destes criarem e desenvolverem repositórios de dados ligados na *Web*, e possibilita também a construção de vocabulários e a escrita de regras para interoperação com esses dados. A ligação (*linking*) de dados é possível com tecnologias como RDF, SPARQL, OWL, SKOS (W3C, 2012).

Segundo a W3C (2012) com o auxílio da *Web* Semântica é possível a utilização e acesso de elementos como:

- Dados ligados (*linked*): informações de dados disponibilizadas para *web* em formato RDF que é a base para a publicação e ligação de dados;
- Vocabulário de ontologias: são utilizados para organizar dados de determinado domínio, a linguagem OWL (*Ontology Web Language*) é utilizada para classificação de dados em domínios específicos;
- Consultas: a *Web* Semântica pode ser visto como uma base global de informações que pode ser consultada e alterada. A linguagem utiliza para a realização das consultas é SPARQL;
- Inferências: informações presentes na *Web* Semântica podem ser inferidas com a utilização de regras e axiomas definidos através de formatos como o RIF (*Rule Interchange Format*) e a linguagem OWL;
- Aplicações Verticais: aplicações da *Web* Semântica nas diferentes áreas do conhecimento com a utilização de domínios específicos;

3.4.1 JENA Framework

O JENA é formado por um conjunto de APIs (*Application Programming Interfaces*) desenvolvidas com a linguagem de programação Java que auxiliam no desenvolvimento de sistemas que utilizam *Web* Semântica. O *framework* disponibiliza funções que permitem realizar leitura e escrita RDF nos formatos XML (*eXtensible Markup Language*), N-triples1 e Turtle2. Também provê bibliotecas de manipulação de ontologias, possibilitando a realização de consultas e inferências sobre na base de dados da *Web* Semântica (JENA, 2012).

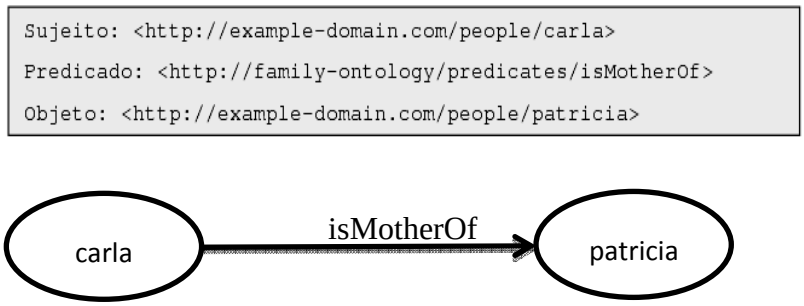
Dentro do projeto JENA surge em destaque as APIs para armazenamento de triplas RDF. Estas ferramentas possibilitam o armazenamento de forma eficiente de uma grande quantidade de informações no formato RDF. O armazenamento das informações em uma base de dados esta disponível a API SDB (*Application Programming Interfaces Suitable DataBase*) para banco de dados relacional, e para o armazenamento no formato de triplas RDF o esta disponível o componente TDB (*Truples DataBase*) (JENA, 2012).

O JENA é uma ferramenta de distribuição gratuita sob a licença Apache, e pode ser encontrado facilmente na página oficial do projeto com uma vasta documentação que auxilia na sua utilização em projetos direcionados a *Web* Semântica (JENA, 2012). O JENA trabalha de forma transparente em relação à linguagem adotada para representação de determinado domínio através de ontologia. De acordo com a linguagem utilizada nas ontologias, a API do JENA habilita ou desabilita determinados recursos, porém, a forma de manipular ou criar uma ontologia é a mesma para qualquer linguagem de representação.

3.4.2 SPARQL

SPARQL é a linguagem de consulta padrão recomendada pelo W3C para recuperação de informações contidas em grafos RDF. Um grafo é constituído a partir de um conjunto de triplas RDF, formando assim uma base de dados semânticos. A representação de uma tripla RDF é ilustrada na Figura 5. As triplas RDF são dados estruturados semanticamente baseados na especificação RDF, estes dados são formatados em sujeito, predicado e objeto.

Figura 5 – Exemplo Codificação Tripla RDF



Fonte: WC3. (2012).

SPARQL é uma linguagem para consulta a base dados estruturados semanticamente, com ela é possível fazer uma analogia entre SPARQL e a linguagem para consultas a bases relacionais SQL (EISENBERG et al, 2004). A linguagem SPARQL possui a estrutura *Select-Form-Where:*

- *Select*: Indica a regra sobre os dados pesquisados, como a ordem e a quantidade de atributos e/ou instâncias que serão retornados.
- *From*: Especifica quais fontes serão consultadas. Esta cláusula é opcional. Quando não especificada, assume-se que a busca será realizada em um documento RDF/RDFS particular.
- *Where*: Aplica restrições na consulta. As informações retornadas pela consulta deverão satisfazer as restrições fixadas por esta cláusula.

Para a utilização direta da linguagem de consultas SPARQL é necessário conhecer sua sintaxe e as ontologias dos domínios específicos armazenados. Pela estruturação do modelo

SPARQL e de como os dados estão representados, atualizações e inserções de novos conjuntos de dados manualmente em ontologias torna-se uma tarefa difícil (W3C, 2012).

SPARQL é uma linguagem de consultas (q*uery*) para à recuperação de informação sobre dados armazenados semanticamente. Por último foi desenvolvido na linguagem a especificação para realização de atualizações (*updates*).

Realizar consultas com SPARQL em base de dados relacionais segue um exemplo, utilizando uma base que possui uma ontologia que descreve o domínio de árvores genealógicas, para realizar uma consulta que retorne o nome dos filhos de uma determinada pessoa, utilizar-se a consulta conforme a figura 6.

Figura 6 – Exemplo de Consulta SPARQL

```
PREFIX family: <http://family-ontology/predicates/>
PREFIX people: <http://example-domain.com/people/>
PREFIX rfds: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?name
WHERE {
  people:carla family:isMotherOf ?daughter .
  ?daughter rfds:label ?name .
}
```

Fonte: WC3. (2012).

Nas primeiras três linhas os prefixos são definidos, para tornar a consulta mais compacta e com uma tendência menor a erros. O termo "*family*:" se torna um sinônimo para "<http://family-ontology/predicates/>" assim como os outros termos também se tornam sinônimos. Na quarta linha é apresentada a definida de consulta, em que a operação de consulta deve retornar o resultado da variável "*name*" (o símbolo interrogação antes de uma palavra em SPARQL indica que se trata de uma variável). No bloco abaixo, é apresentada a sintaxe "*WHERE*", na próxima linha, as restrições para a execução da consulta são apresentados. Dentro das restrições de consulta, na primeira linha é definido que existe uma variável chamada "*daughter*", e esta variável possui o valor de todos os recursos que estão na posição OBJETO em uma tripla RDF, e que esta tripla tenha "*people*:carla" como o SUJEITO e "*family:isMotherOf* " como PREDICADO. Na segunda restrição, são fixados os recursos da variável "*daughter*" como SUJEITO, o PREDICADO é definido como "*rfds:label*" e também define o valor da variável "*name*", que contém o retorno da consulta. Executando essa consulta em uma base de dados RDF que possua as triplas apresentadas, o retorno da consulta seria um nome próprio descrito na ontologia com OBJETO.

3.5 Mecanismos de Inferências

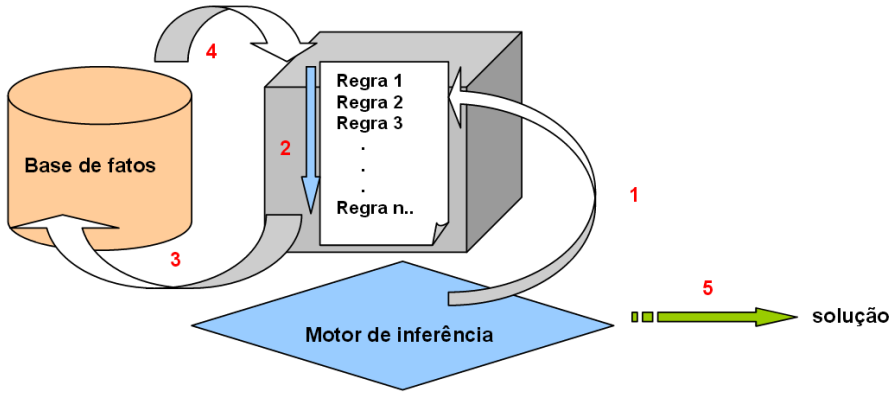
Um mecanismo ou motor de inferência é formado por um sistema que executa operações baseadas em regras de inferência ou de dedução sobre uma base de conhecimentos, sendo capaz de extrair conclusões de forma automática com o objetivo de encontrar uma solução para um problema. Em um motor de inferência devem estar presentes funcionalidades como: modo de raciocínio, estratégia de busca, resolução de conflitos e representação de incertezas.

Nos motores de inferência existem dois modos de raciocínio que podem ser aplicados as regras de inferência: encadeamento progressivo (*forward chaining*), e o encadeamento regressivo (*backward chaining*) (GRIFFIN, LEWIS. 2013). No encadeamento progressivo, também conhecido como encadeamento dirigido por dados, a parte esquerda da regra é comparada com a descrição da situação atual, contida na memória de trabalho. Quando uma regra satisfaz a condição da esquerda, então a parte direita da regra é executada, o que, em geral significa a introdução de novos fatos na memória de trabalho. No encadeamento regressivo, conhecido também como encadeamento dirigido por objetivos, o comportamento do sistema deve ser controlado por uma lista de objetivos. Um objetivo pode ser satisfeito por um elemento da memória de trabalho, ou pela existência de regras que permitem a inferência deste objetivo. As regras que satisfazem esta condição têm as instâncias correspondentes as suas partes esquerdas adicionadas a lista de objetivos correntes. Quando uma destas regras tenha todas as suas condições satisfeitas diretamente pela memória de trabalho, o objetivo em sua parte direita também é adicionado a memória de trabalho. Um objetivo que não possa ser satisfeito diretamente pela memória de trabalho, nem inferido através de uma regra, é abandonado. Quando o objetivo inicial é satisfeito, ou não há mais objetivos, o processamento é cessado.

A estratégia de busca é utilizada para guiar a pesquisa na memória trabalho e na base de regras. A resolução de conflitos é executada após o término da busca quando o motor de inferência possui um conjunto de regras que satisfazem as premissas atuais do problema, este conjunto recebe o nome de conjunto de conflito. Então o motor de inferência deve escolhe as regras que serão executadas e a ordem da execução destas regras. A quantificação de incertezas são atribuídos valores de pesos quantitativos aos fatos e as regras.

A figura 7 mostra o processo de operação realizado pelo motor de inferências, do recebimento da solicitação para percorrer a base de regras, gravar novos fatos e enviar resposta final.

Figura 7 – Fluxo de Operações no Motor de Inferência



Fonte: GRIFFIN, LEWIS. (2013).

Existe uma grande quantidade de ferramentas utilizadas em raciocínio lógico, estas ferramentas vão de aplicações utilizadas externamente nas aplicações até bibliotecas adicionadas ao sistema desenvolvido rodando em tempo de execução. Dentre estas ferramentas são destacadas o PELLET (SIRIN et al, 2007), JESS (JESS, 2013), JBOSS Drools (DROOLS, 2013), JRuleEngine (JRULEENGINE, 2013). Nesta dissertação são listados alguns mecanismos de inferência para análise de operação e futuras aplicações.

3.5.1 PELLET

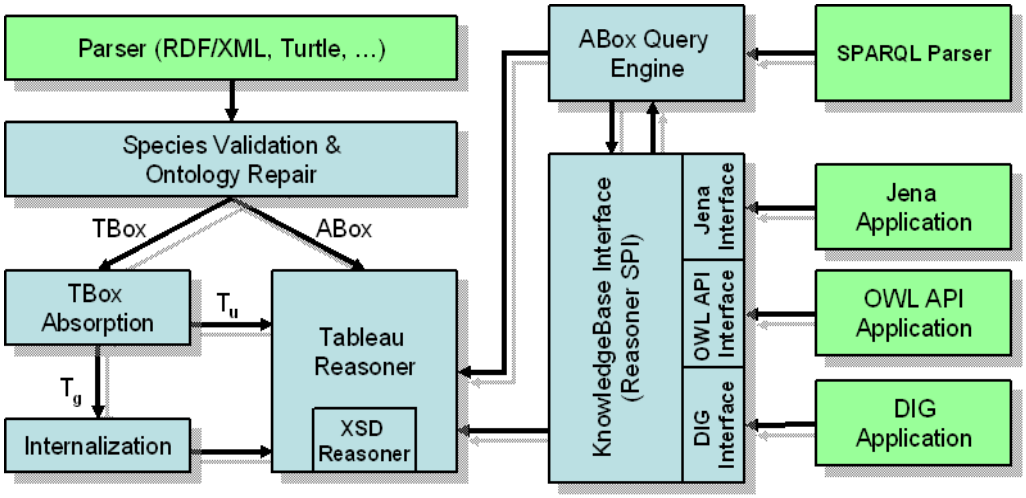
Pellet (SIRIN et al, 2007) é um *reasoner* de código aberto escrito em linguagem de programação Java para o motor de inferência OWL-DL (*Web Ontology Language – Description Logic)* e disponibiliza controle de raciocínio para a lógica de descrição SHOIQ. O Pellet é baseado em algoritmos *Tableaux* para lógicas de descrição expressivas e pode ser utilizada em conjunto com o Jena. Com este *reasoner* pode-se utilizar os principais serviços de inferência para lógicas de descrição como: checagem de consistência, nível de satisfação, classificação e realização.

Dentre algumas características do Pellet pode-se citar a portabilidade, facilidade de integração com os principais frameworks e API´s para desenvolvimento de ontologia, suporte as regras SWRL (*Semantic Web Rule Language*). Além disso, o Pellet está em constante aprimoramento técnico e conta também com a participação ativa dos usuários até mesmo devido a seu caráter open-source (SIRIN et al, 2007).

O Pellet disponibiliza um mecanismo para execução de *queryes* sobre a base de dados *ABox* do conhecimento. Para isso é utilizado a linguagem de SPARQL (W3C, 2012). O SPARQL é utilizado para a consulta de dados para o padrão RDF, no qual a linguagem OWL é baseada.

A figura 8 mostra os principais componentes do Pellet. Na qual, o Pellet é um motor de inferência para DL (*Description Logic*). De maneira que os serviços de inferência são reduzidos a verificações de consistência. O mecanismo permite a utilização de outros algoritmos do tipo *tableaux* sejam utilizados. O algoritmo padrão trabalha com SROIQ(D), porém existem outros tipos de algoritmos, por exemplo, para extensões não-monotônicas e integração com regras.

Figura 8 – Arquitetura do Pellet



Fonte: SIRIN et al, (2007).

O núcleo do Pellet é o mecanismo de inferência (*Tableaux reasoner*), em que este *reasoner* verifica a consistência de uma base de conhecimento. O mecanismo de inferência possui um módulo que trabalha com tipos de dados, responsável por verificar a consistência de conjunções do tipo XML-S. As ontologias OWL são carregadas pelo mecanismo após a validação das classes e o acerto de inconsistências na ontologia *(Species Validation & Ontology Repair*). A execução desta etapa assegura que todos os recursos tenham uma tripla

de tipo apropriada, e declarações de tipos inexistentes sejam adicionadas de acordo com uma heurística.

Na fase de carga da ontologia, os axiomas das classes são carregados pelo componente *TBox* e as asserções próximas dos indivíduos são armazenadas no componente *ABox*. Axiomas *TBox* passam pelo pré-processamento de normalização, absorção e internalização. Antes de serem analisadas pelo mecanismo *tableaux*. O sistema fornece uma camada para o acesso aos serviços de inferência através de uma *Service Programming Interface* (SPI). A execução da tarefa de consultas permite responder consultas ABox expressas em SPARQL ou RDQL.

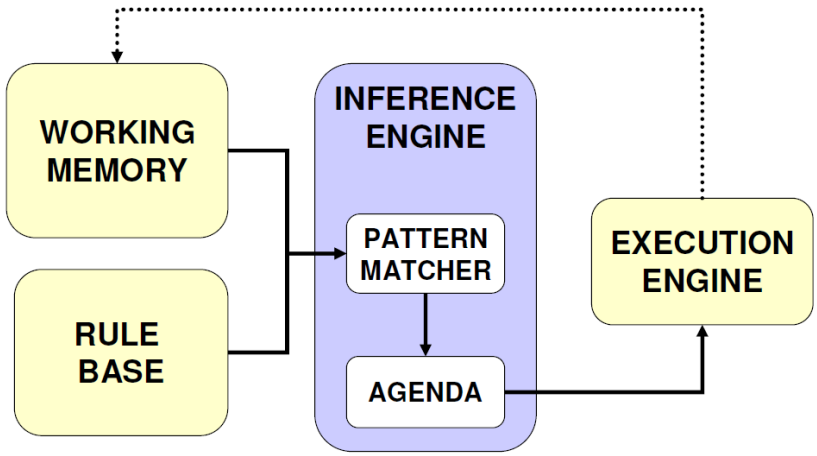
3.5.2 Java Expert System Shell (JESS)

O JESS foi criado e desenvolvido por *Ernest Friedmal-Hill* no *Sandia National Labs* como parte de um projeto interno de pesquisa. Sua primeira versão foi publicada em 1995. O JESS é um *script shell* para construção de sistemas especialistas baseado em regras inteiramente desenvolvido em Java. Possui um ambiente similar a linguagem de programação CLIPS (*C Language Integrated Production System*) desenvolvida pela NASA. O JESS é distribuído sob dois tipos de licença, uma paga, e outra para desenvolvimento acadêmico com licença para 365 dias (JESS, 2013). O JESS utiliza o algoritmo RETE na sua máquina de inferência. O algoritmo RETE é caracterizado por organizar as regras em uma árvore, de modo que sejam divididas em padrões, assim, regras semelhantes percorrerão o mesmo ramo da árvore enquanto possuírem cláusulas iguais.

A linguagem de programação utilizada no JESS pode acessar diretamente todas as classes e bibliotecas Java. Por esta razão, é uma ferramenta muito utilizada como ambiente de desenvolvimento rápido e de aplicações de *scripts* (JESS, 2013).

O JESS é um sistema especialista baseado em regras. Este sistema consiste de uma base de regras, uma base de fatos e do motor de inferência como mostrado na figura 9, a motor de inferência controla todo o processo de aplicação de regras a memória de trabalho para obter a saída do sistema. Estas regras podem inferir novos fatos que serão adicionados a base de fatos ou podem simplesmente executar funções desenvolvidas em Java (JESS, 2012).

Figura 9 – Diagrama da Arquitetura JESS



Fonte: JESS. (2012).

O sistema especialista desenvolvido em JESS possui suas regras no formato de construção do tipo *se...então* semelhante as linguagens de programação convencionais. Sendo que em programas convencionais as construções *se...então* são executadas em uma ordem específica de acordo com a sequência em que o desenvolvedor as programou, no JESS, quando se utiliza o encadeamento progressivo (*forward chaining*), as regras são executadas sempre que sua parte da direita referente ao "*se*" seja satisfeita, e este condição é executada desde que o motor de inferência esteja rodando. Isto torna as regras do JESS menos determinísticas que os programas convencionais (JESS, 2013).

Uma regra escrita em JESS possui a seguinte sintaxe:

```
(defrule disjuntor-fonte-desligado
    (object (is-a    dj-fonte)(estado FALSE))
    =>
    (assert (sinal renovar))
    (bind ?*topologia* ""))
```

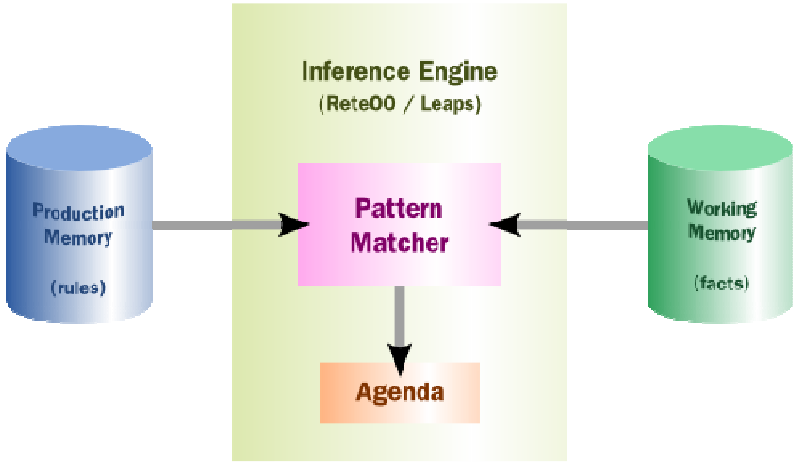
No exemplo listado acima é possível ver que a regra é composta do nome "disjuntor-fonte-desligado" seguido da condição de encadeamento progressivo (*forward chaining*), que fica à esquerda da seta. No exemplo acima, será satisfeito se for encontrado na base de fatos um objeto do tipo "dj-fonte" cujo seu atributo estado esteja no estado lógico *FALSE*. Neste caso, a ação que se segue quando executar a regra é a criação na base de fatos de um novo fato (sinal renovar) e do armazenamento do caractere vazio " " na variável ?*topologia*.

3.5.3 JBOSS Drools

É um *shell* para sistemas especialistas baseado em regras de negócio *business rule management system* (BRMs). Utilizando para sua inferência o processo de encadeamento progressivo, encadeamento regressivo, ou regras de produção, utilizando o a aplicação do algoritmo RETE para a linguagem de programação Java no seu aprimoramento (DROOLS, 2013). JBoss possui uma plataforma de lógica de negócio chamada JBoss Drools, que possui um de motor de inferência. O sistema teve sua primeira versão lançada em 2001, estando atualmente na versão 6.0.0.CR3.

O Drools é composto basicamente por uma máquina de inferência responsável pela execução das regras, de uma memoria de trabalho que armazena os fatos gerados pela execução das regras, de uma base de conhecimento é o local onde estão contidas as regras a serem utilizadas pelo mecanismo de inferência. Na figura 10 é mostrado o diagrama dos elementos que compõe o *shell* Drools.

Figura 10 – Diagrama da Arquitetura JBOSS Drools



Fonte: Drools. (2013).

Com o Drools é possível elaborar regras de negócio declarativamente, separar e centralizar as regras de negócio de uma aplicação, e gerenciar regras alterando-as e alterando suas versões dinamicamente. Além disso, é uma plataforma desenvolvida pensando-se em no desempenho e escalabilidade do sistema.

Na figura 11 é mostrado um exemplo de duas regras utilizadas no sistema Drools. Nestas regras, o mecanismo de inferência executa a primeira regra, quando o sistema receber a mensagem "*Hello Word*", o sistema mostra a mensagem no *display* e seta o seu estado para "*Goodbye*" e altera a mensagem recebida para "*Goodbye cruel world*". Na próxima execução do mecanismo de inferência a regra executada será a regra com nome "*GoodBye*" que irá mostrar no *display* a mensagem que foi alterada na regra anterior.

Figura 11 – Exemplo de regra utilizada em JBOSS Drools

```
package com.sample

import com.sample.DroolsTest.Message;

rule "Hello World"
    when
        m : Message( status == Message.HELLO, myMessage : message )
    then
        System.out.println( myMessage );
        m.setMessage( "Goodbye cruel world" );
        m.setStatus( Message.GOODBYE );
        update( m );
end

rule "GoodBye"
    when
        Message( status == Message.GOODBYE, myMessage : message )
    then
        System.out.println( myMessage );
end
```

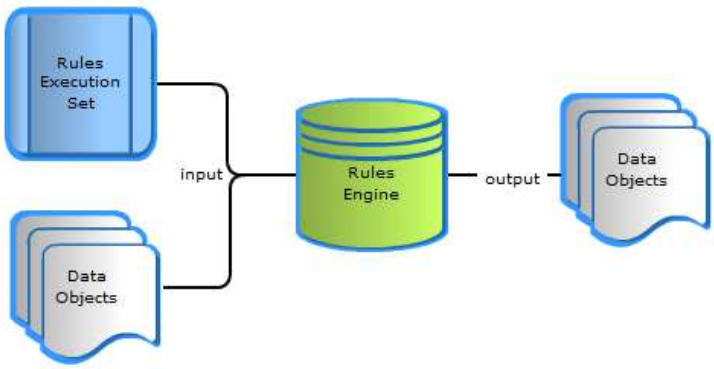
Fonte: Drools. (2013).

3.5.4 JRuleEngine

JRuleEngine é um mecanismo de inferência JAVA, desenvolvido com base na especificação *Java Specification Request* 94, versão 1.1 (JSR-94). No mecanismo de inferência do *JRuleEngine* os objetos de entrada são referidos como fatos e objetos de saída são referidos como conclusões. Os métodos de determinada classe, podem ser chamados diretamente a partir de suas regras. O algoritmo utilizado na criação deste mecanismo de inferência é baseado em encadeamento progressivo (*JRUELENGINE*, 2013).

A arquitetura do *JRuleEngine* é composta de uma memória de trabalho onde estão contidas os fatos e fatos gerados da execução das regras do sistema, uma base de conhecimento onde as regras para inferência estão armazenadas, e o mecanismo de inferência responsável pelo processamento das regras, acesso a métodos externos e alimentação da base de fatos, conforme mostrado na figura 12.

Figura 12 – Diagrama da Arquitetura JRuleEngine



Fonte: JRuleEngine. (2013).

As regras do sistema podem desenvolvidas em duas formas distintas, de arquivos no formato *xml* especifico do *JRuleEngine*. Ou por objetos do tipo *RuleImpl* que podem ser recuperados de uma base de dados. Existem dois tipos distintos de regras no sistema, as regras de sessão do tipo *statefull* que são executadas por um período longo de tempo e podem ser consultadas novamente dentro da mesma execução, e as regras do tipo *stateless*, estas proporcionam um melhor desempenho ao sistema com curta duração de tempo para sua execução.

Na figura 13 é mostrado um exemplo de regra no formato *xml* utilizado no mecanismo de inferência *JRuleEngine*. Neste exemplo de regra, ocorre o seguinte: se o parâmetro ":X" for igual a "*is human*" o mecanismo guarda na memoria de fatos o novo elemento através de um método externo, e seta duas variáveis externas utilizados na impressão no *display* do processamento da regra.

Figura 13 – Exemplo de regra xml utilizada no JRuleEngine

```
<?xml version="1.0" encoding="UTF-8"?>
<rule-execution-set>
  <name>RuleExecutionSet1</name>
  <description>Rule Execution Set</description>

  <synonymn name="prop" class="org.jruleengine.Clause" />

  <!--
  if :X is human then :X is mortal
  -->
  <rule name="Rule1" description="if :X is human then :X is mortal" >
   <if leftTerm=":X" op="=" rightTerm="is human" />
   <then method="prop.setClause" arg1=":X" arg2="is mortal" />
  </rule>

</rule-execution-set>
```

Fonte: JRuleEngine. (2013).

3.6 Objetos de Aprendizagem

Um Objeto de Aprendizagem (OA) é qualquer material digital (ou até mesmo não digital), que possa ser usado, reutilizado e referenciado, através de meios tecnológicos, no suporte ao aprendizado (IEEE-LTSC 2002). A reutilização é um aspecto importante destes objetos (WILEY, 2001).

Para que um recurso digital seja considerado um OA, ele deve apresentar características que a identifique como tal. Desta forma, os recursos que formam os OA podem ser analisados em dois aspectos: pedagógico e técnico.

Segundo Dias (2009), as características pedagógicas estão relacionadas com a criação de objetos que facilitem a mediação de professores e alunos, visando à aquisição do conhecimento:

(1) Interatividade: recursos de aprendizagem que oferecem suporte às concretizações e ações.

(2) Autonomia: recursos que propiciem a iniciativa e tomada de decisão.

(3) Cooperação: usuários podem trocar ideias e trabalhar coletivamente sobre o conceito apresentado.

(4) Cognição: referente às sobrecargas cognitivas colocadas ao aprendiz durante a instrução;

(5) Afetividade: relacionado aos sentimentos e motivações do aluno com sua aprendizagem e colegas.

Já as características técnicas referem-se, a questões de padronização, classificação, armazenamento, recuperação, transmissão e reutilização dos objetos de aprendizagem. São características técnicas específicas dos Objetos de Aprendizagem (DIAS, 2009):

(1) Acesso: indica se um OA pode ser utilizado localmente ou, por apenas um ou por mais de um acesso.

(2) Agregação: indica quais recursos podem ser reunidos em conjuntos maiores de conteúdos, incluindo estruturas tradicionais de cursos.

(3) Autonomia: verifica se o objeto pode ser usado de forma autônoma, ou depende de outros fatores.

(4) Classificação: há uma catalogação dos objetos que auxilie na identificação dos mesmos e facilite o trabalho de busca e recuperação de conteúdo.

(5) Formato: qual o formato digital do conteúdo;

(6) Durabilidade: indica se a utilização dos recursos educacionais se mantém quando a base tecnológica muda, sem a necessidade de um novo projeto ou recodificação;

(7) Interoperabilidade: envolve em utilizar os OA em diferentes locais, independente de ferramentas ou plataformas.

(8) Reutilização: que pode variar de acordo com a granularidade do Objeto de Aprendizagem.

Os aspectos técnicos dos OA são influenciados por questões de padronização. As funcionalidades como acesso, agregação, reutilização, interoperabilidade, além do intercâmbio de informações entre os diversos tipos de formatos digitais, entre outras características dos OA, são diretamente dependentes da existência de padrões para se tornarem possíveis. A reutilização de conteúdo, que consiste em uma forma eficiente de readaptar o conteúdo dos Objetos de Aprendizagem para diferentes tipos de contextos e usuários, somente se torna possível com o auxílio dos padrões. Desta forma, surgiram algumas iniciativas visando padronizar a especificação, construção e a identificação dos Objetos de Aprendizagem, através da adoção de modelos e padrões para o desenvolvimento destes (DIAS, 2009).

Seguindo este contexto, podem-se citar organizações e grupos de pesquisas que vêm trabalhando para construir e aperfeiçoar a eficiência e eficácia dos objetos de aprendizagem, concentrando-se principalmente na definição de padrões. São eles*: Learning Technology Standard Comitee* (LTSC) do *Institute of Electrical and Electronics Engineers* (IEEE) (IEEE-LTSC 2002), a *Alliance of Remote Institute of Electrical and Distribution Networks for Europe* (ARIADNE), o IMS *Global Learning Consortium*, a *Canadiam Core e a Advanced Distributed Learning Initiative* que têm contribuído significativamente na definição de padrões de indexação (metadados) (KRATZ; CRESPO; BARBOSA, 2007).

No caso das padronizações dos objetos de aprendizagem existem diversas recomendações definidas, a respeito dos metadados que catalogam e que caracterizam os Objetos de Aprendizagem, bem como também, recomendações de como estes Objetos de Aprendizagem podem ser encapsulados e como seus conteúdos podem ser navegados (W3C, 2012). Em uma lista podem ser citados como principais tipos: O LOM (*Learning Object Metadata*) do *IEEE Learning Technology Standards Committee* (LTSC), DCMI *(Dublin Core Metadata Initiative*), SCORM (*Sharable Content Object Reference Model*) ADL (ADL, 2010) e IMS do *Global Learning Design* (DUTRA e TAROUCO, 2010).

O desenvolvimento de uma série de metadados para Objetos de Aprendizagem torna a descrição dos objetos uma atividade complexa e que dependente de um vasto conhecimento sobre os padrões de metadados existentes. A grande quantidade de padrões existente dificulta

a comunicação entre sistemas diferentes e faz com que a criação de objetos de aprendizagem adaptáveis a múltiplas plataformas e formatos seja uma atividade humanamente inviável.

Nos próximos itens serão listados e apresentados alguns padrões com maior relevância para a pesquisa. Dentre os padrões apresentados podem-se citar três padrões estudados referentes ao conceito de metadados: Dublin Core, IEEE-LOM e OBAA, e mais um padrão para encapsulamento do conteúdo de metadados: SCORM. Todos estes padrões são utilizados atualmente e o conhecimento de todos estes padrões é importante no processo de reutilização de Objetos de Aprendizagem em diversos seguimentos e áreas tecnológicas.

3.6.1 Dublin Core

É um padrão desenvolvido pela DCMI (*Dublin Core Metadata Initiative*). Que é uma instituição dedicada a promover a adoção de padrões para a interoperabilidade de metadados e desenvolver vocabulários especializados para descrever bases de informações que tornem mais inteligentes sistemas especialistas em busca de informações.

A DCMI desenvolve atividade que incluem trabalhos relacionados a arquitetura e a modelagem do padrão referido como Dublin Core. Sendo apresentado como um fórum aberto, a DCMI convida e pede a pesquisadores e utilizadores de conteúdo para que contribuam com o projeto. O padrão Dublin Core foi desenvolvido pela DCMI inicialmente para facilitar a pesquisa e a recuperação de objetos na *Web*. Em 2003 o Dublin Core foi aprovado como uma norma da ISO (*International Standard Organization*) e atualmente mantém a versão corrente 1.1 com 15 metadados bem definidos, são eles (DCMI, 2008):

- *Title*: título do objeto;
- *Creator*: identificação do criador do objeto;
- *Subject*: assunto sobre o qual o objeto se refere;
- *Description*: texto com uma descrição das características do objeto;
- *Publisher*: identificação do publicador;
- *Contributor*: identificação do contribuidor;
- *Date*: data de criação do objeto;
- *Type*: tipo do objeto;
- *Format*: formato do conteúdo;
- *Identifier*: identificador único para o objeto;
- *Source*: recurso de onde este originou;
- *Language*: idioma em que o conteúdo está escrito;
- *Relation*: relações com outros objetos;
- *Coverage*: abrangência do objeto, assuntos ou tópicos tratados pelo objeto;
- *Rights*: direitos autorais (ou outros) referentes ao objeto e seu conteúdo.

Este padrão é utilizado para a busca de objetos digitais em diversos assuntos, e fornece suporte a catalogação em locais como museus, bibliotecas, instituições de pesquisa e *softwares* diversos.

Os metadados deste padrão não possuem obrigatoriedade em seu preenchimento e podem ser repetidos dentro de uma mesma descrição garantindo assim uma maior flexibilidade. O padrão Dublin Core começou como um subconjunto de metadados básicos, chamado de Dublin Core Simples. Com o decorrer de seu desenvolvimento este formato ele recebeu extensões que são chamadas Dublin Core Qualificado.

3.6.2 IEEE-LOM

Criado oficialmente em junho de 2002 o padrão LOM (*Learn Object Metadata*) é promovido pela IEEE. O padrão de metadados IEEE-LOM possui uma estrutura voltada a conteúdos relativos a Objetos de Aprendizagem. Este padrão define mais de 70 atributos que estão agrupados em nove categorias, são elas (IEEE, 2002):

- *General*: informações gerais sobre o Objeto de Aprendizagem;
- *Lifecycle*: informações sobre o histórico e estado atual do Objeto de Aprendizagem;
- *Meta-metadata*: informações sobre os metadados em si;
- *Technical*: características e requisitos técnicos do Objeto de Aprendizagem;
- *Educational*: características educacionais e pedagógicas do Objeto de Aprendizagem;
- *Rights*: informações sobre propriedade intelectual do Objeto de Aprendizagem;
- *Relation*: define os relacionamentos entre este Objeto de Aprendizagem; e outros Objetos de Aprendizagem;
- *Annotation*: comentários sobre o uso educacional do Objeto de Aprendizagem;
- *Classification*: informações de classificação do objeto dentro de sistemas de classificação (taxonomias);

O padrão de metadados IEEE-LOM possui partes opcionais propiciando seu preenchimento flexível para cada caso. Alguns dos metadados descritos pelo padrão IEEE-LOM possuem uma lista de termos permitidos, facilitando assim a automatização e a validação do processo de preenchimento.

Apesar do IEEE-LOM possuir uma variedade imensa de documentação, e esta documentação ser de fácil acesso, a quantidade de metadados a ser preenchida pode se tornar um desafio a usuários iniciantes devido à curva de aprendizado desta documentação (MCCLELLAND, 2003).

3.6.3 OBAA – Objetos de Aprendizagem baseado em Agentes

A proposta de metadados Objetos de Aprendizagem baseado em Agentes (OBAA) tem como objetivo permitir a interoperabilidade entre as tecnologias de Objetos de Aprendizagem nas diversas plataformas tecnológicas existentes. Com o resultado da proposta foi definido um padrão de metadados aberto que é compatível com os padrões de metadados existentes no mercado atualmente: IEEE-LOM, Dublin Core (VICCARI et al., 2009).

Os principais quesitos da proposta para metadados OBAA são os seguintes:

(1) Padrão aberto e flexível;

(2) Compatibilidade com padrões atuais de metadados de Objetos de Aprendizagem, em particular com IEEE-LOM e Dublin Core;

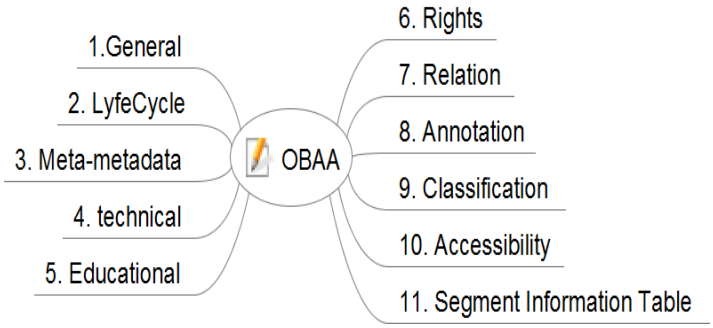
(3) Suporte as questões educacionais do Brasil;

(4) Suporte a interoperabilidade com plataformas *Web*, TV Digital (SBTVD) e dispositivos móveis;

(5) Suporte a requisitos de acessibilidade digital;

O padrão OBAA especifica um conjunto de metadados que permitem a catalogação de Objetos de Aprendizagem (ver Figura 18). Estes metadados foram propostos para especificar uma ampla gama de funcionalidades possibilitando a interoperabilidade em diversas plataformas (VICCARI et al., 2009).

Figura 14 - Metadados OBAA.



Fonte autor (2012).

Os metadados OBAA possuem uma extensão dos metadados IEEE-LOM, formada pela adição de novos grupos de metadados:

- Metadados técnicos multiplataforma: extensão do grupo quatro de metadados técnicos do IEEE-LOM;
- Metadados pedagógicos: extensão dos metadados educacionais do IEEE-LOM;
- Metadados de acessibilidade: novo de metadados para acessibilidade, grupo criado para descrever requisitos de pessoas com necessidades especiais, adaptados do padrão IMS AccessForAll;
- Metadados de segmentação multimídia: novo grupo de metadados para informações de segmentação de conteúdos multimídia, compatíveis com o padrão MPEG-7.

Os metadados OBAA o não precisam ser utilizados em sua totalidade de informações, mas apenas os metadados necessários para sua aplicação ou finalidade, isto é obtido, através da utilização de perfis que englobam as principais aplicações para os objetos.

Para chegar ao objetivo de compatibilidade e ao mesmo tempo manter principais características dos padrões de metadados existentes, um grande estudo foi desenvolvido (VICCARI et al., 2009). Neste foi estudo levado em consideração características como a abrangência para maximizar a reutilização e a adaptabilidade para as diversas plataformas. A abrangência em conteúdos educacionais do padrão IEEE-LOM é mais extensa em comparação ao padrão Dublin Core, desta maneira o padrão IEEE-LOM é a base para o desenvolvimento das especificações educacionais do padrão OBAA. Quanto a abrangência do padrão, a analise é em relação às características de interoperabilidade entre plataformas, e o padrão MPEG-7 é muito mais utilizado para conteúdos multimídia que o IEEE-LOM.

3.7 Ontologia

A palavra ontologia é deriva do Grego onde; ontos (ser) + logos (palavra). Fora introduzida na filosofia no sec. XIX, por filósofos Alemães. Sendo uma disciplina filósofa, a ontologia tem como objetivo catalogar as diferentes visões do que representa o mundo (Guarino 1998).

Hoje existem muitas definições para o termo ontologia, estes podem ser encontrados em literaturas técnicas e na internet. Estas definições apresentam pontos de vista diferentes e complementares sobre o assunto. Há autores que expõem definições independentes do processo utilizado na construção da ontologia, e outros autores orientam fortemente a utilização do processo de desenvolvimento tradicional de uma ontologia.

Uma ontologia pode ter formas variadas, mas deverá obrigatoriamente incluir um vocabulário de termos e a especificação dos seus significados (USCHOLD, JASPER, 1999). Isto inclui a definição e a indicação de como os conceitos estão relacionados entre si, o que infunde uma estrutura ao domínio e restringe a possibilidade de interpretação dos termos (GÓMEZ, PEREZ, 2004).

Embora não se tenha um consenso em relação a uma única definição para o termo ontologias, pode-se considerar que ontologias visam capturar o conhecimento consensual de forma genérica, e este conhecimento pode ser reutilizado e compartilhado entre aplicações e grupos de pessoas. Em geral ontologias são construídas cooperativamente por diferentes grupos de pessoas, em diferentes locais (GÓMEZ, PEREZ, 2004).

Na tecnologia de informação, as ontologias foram desenvolvidas na área da Inteligência Artificial (IA) para facilitar a reutilização e a compartilhamento de informações. Atualmente, as ontologias são largamente utilizadas em áreas como a integração de informação inteligente, comércio electrónico, engenharia de *software* baseado em agentes, entre outras (BREITMAN, LEITE, 2003).

Nos temas da literatura na área de Inteligência Artificial existem diversas definições sobre ontologia. Segundo (NOY, MCGUINNESS, 2001), uma ontologia é a descrição explícita e formal de conceitos em um determinado domínio. Esta descrição é então composta por classes (também chamadas de conceitos), propriedades de cada classe descrevendo as características e atributos da classe. É possível criar diversas instâncias para cada uma das classes (indivíduos). As instâncias em juntamente com as classes formam a base de conhecimento. Existe a possibilidade de, as classes conterem subclasses que representam conceitos mais específicos destas classes.

A modelagem de ontologia utilizada atualmente é baseada em noções de Frames e na semântica das Lógicas Descritivas (DL – *Description Logics*) (GÓMEZ-PEREZ,2004). Os dois conceitos apresentam semelhanças em seu modelo de construção: ambos são construídos utilizando a noção de classes, que representam conceitos de determinado domínio. Classes possuem instâncias; propriedades (*slots*) que descrevem atributos dessas classes e o relacionamento entre elas; suas restrições sobre os valores das propriedades/*slots*. Existem, entretanto, muitas diferenças na semântica dessas construções e na forma como elas são utilizadas para a inferência de novos fatos a partir da ontologia, ou para determinar se a ontologia é consistente. O conceito atualmente utilizado para definir a semântica de uma ontologia é a linguagem OWL (*Web Ontology Language*) definida como o padrão para especificação de ontologias pelo W3C (*World Wide Web Consortium*) (W3C, 2011).

4 TRABALHOS RELACIONADOS

A catalogação de conteúdo de OA é o objetivo central desta dissertação. Para que o processo de catalogação seja possível é importante analisar a literatura em busca de trabalhos sobre esse tipo de processo. Desta forma neste capítulo serão analisados e apresentados trabalhos relacionados ao tema da pesquisa.

Na literatura foram encontrados vários trabalhos e ferramentas de apoio ao processo de autoria de OA, incluindo em alguns casos a edição de metadados entre as funcionalidades das ferramentas.

Dentre os exemplos encontrados pode-se citar: ALOHA, Aprendaris.cl, Atenex, CourseLab, Cyclone, eXe Learning, FreeLOms, Lectora, LOMPad, MySCORM, PALOMA, RELOAD, Rustici, UniSAnet, Xerte, e DSPACE.

Porém a grande parte dessas ferramentas, como é o caso de ALOHA, Cyclone e UniSAnet, não tiveram continuidade, não estando mais ativas ou disponíveis. Outras ferramentas, como Aprendaris.cl, Atenex, CourseLab, FreeLoms e Lectora são basicamente editores de conteúdo, não estando diretamente relacionado aos objetivos da presente pesquisa.

Assim foram selecionados para uma análise mais detalhada as ferramentas eXe Learning, Xerte e DSPACE, vistas a seguir.

4.1 eXe Learning

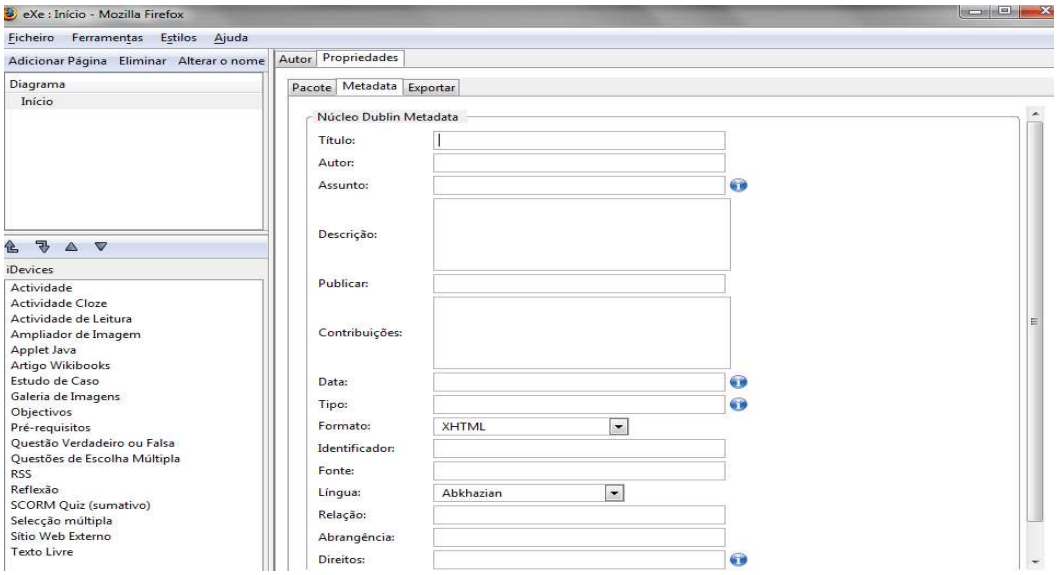
O eXe Learning (http://exelearning.org/wiki#) (BARBONE e RIFON, 2009) é uma ferramenta criada com a linguagem de programação *Python* com a utilização de recursos do *Mozilla FireFox* sendo distribuída sob licença *opengl* de código aberto. É uma ferramenta destinada a autoria de OA, com recursos para auxiliar na construção destes objetos de aprendizagem, permitindo a adição de textos, personalização do objeto com questões, e componentes externos (*applets* e animações *Flash*, imagem, documentos). O *software* permite a exportação de objetos de aprendizagem para as especificações IMS *Content Package* e SCORM 1.2 (ADL, 2010), os metadados destes objetos estão no formato Dublin Core (KUNZE e BAKER, 2007).

Esta ferramenta possibilita estruturar o seu conteúdo em Tópico, Seção e Unidade, com um *link* como raiz, chamado Início. O nome das estruturas pode ser alterado para qualquer nome que se deseje. A esquerda da interface do *software* visualiza-se a estrutura do conteúdo, e a lista dos *iDevices* que serão utilizados pelo desenvolvedor de Objetos de Aprendizagem. O modo edição é apresentado em abas para o preenchimento do formulário com alguns valores para metadados dos objetos como título, nome do autor, língua e descrição.

A criação do catálogo de metadados na ferramenta é constituída de forma manual em dois níveis, no primeiro nível devem ser relacionadas informações do tipo meta-metadada, como nome do autor, título do objeto, e observações importantes sobre o objeto. No segundo nível mostrado na figura 14, o usuário deve realizar o preenchimento dos metadados básicos do padrão Dublin Core (DCMI, 2008), como título, autor, assunto, descrição, formato, fonte, idioma, abrangência, data, direitos, relação, tipo, contribuições, publicar, e identificador.

Pela característica da ferramenta em autoria de conteúdo, esta não possui um catalogo de metadados apropriado ao domínio do objeto de aprendizagem que esta sendo criado pelo usuário. Seus metadados são estruturados para atender qualquer tipo e formatos de Objeto de Aprendizagem de forma simples e genérica.

Figura 15 – Metadados eXe Learning



Fonte: eXe Learning (2012).

Seu editor de conteúdo é um editor de texto com algumas funcionalidades adicionais. O pacote criado pelo *software* é gravado em um arquivo em formato próprio do eXe Learning.

A ferramenta possibilita a criação de objetos de aprendizagem, denominados no contexto do eXe learning de *iDevices*, que são dispositivos instrucionais disponíveis para o desenvolvedor de OA. A criação de novos *iDevices* é possível através do editor de *iDevices* disponível na ferramenta.

4.2 Xerte

O Xerte (http://www.nottingham.ac.uk/xerte/) (BALL e TENNEY,2008) é uma ferramenta distribuída sob a licença GNU *public license*, está incorporada em um projeto que visa o desenvolvimento de OA ricos em conteúdos de aprendizagem através do uso de recursos tecnológicos da plataforma *Flash*, utilizada para criar os Objetos de Aprendizagem. Esta ferramenta suporta diversos tipos conteúdo como texto, imagens e vídeos, permite também que seja integrado ao conteúdo, vídeos do *youtube*, mapas do *google maps* e também artigos como os da *Wikipédia*. O conteúdo pode ser apresentado de duas formas: em tópicos ou sequencial.

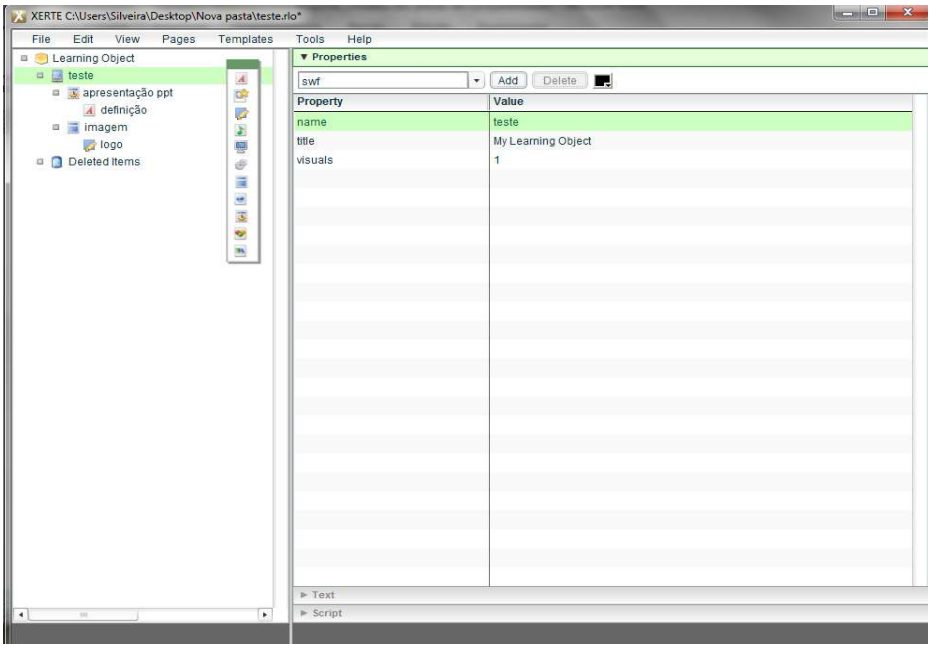
A ferramenta Xerte, oferece um editor *XML* e uma ferramenta de execução que facilita a autoria de objetos de aprendizagem interativos e em conformidade a especificação SCORM 1.2 (ADL, 2010).

Para a autoria de conteúdos no Xerte é necessário ter um conhecimento básico em programação *HTML*, já para a autoria de conteúdos interativos mais complexos, como

calcular a pontuação de um jogo de perguntas e respostas, o usuário necessita de conhecimentos de programação na linguagem *ActionScript* (linguagem utilizada para a criação de *Rich Internet Applications)* (Adobe Systems Incorporated, 2013). Para os desenvolvedores de Objeto de Aprendizagem menos experientes em programação, a ferramenta disponibiliza um assistente para criação de conteúdos, em que o desenvolvedor poderá criar conteúdos de maneira interativa, com a configuração mínima para sua publicação.

A criação do catálogo de metadados para a ferramenta Xerte, é de forma simples e genérica utilizando apenas metadados básicos de identificação do objeto. Os metadados solicitados pelo aplicativo são informações básicas de identificação do Objeto de Aprendizagem que esta sendo utilizado para a nova autoria, como mostrado na figura 15. As informações são: nome, descrição do objeto em texto livre, localização.

Figura 16 – Exemplo Metadados Xerte



Fonte: Xerte (2012).

A nível meta-metadata o Xerte utiliza as informações recebidas no momento da criação do novo Objeto de Aprendizagem, estas informações apenas identificam o objeto como nome, autor e localização para a publicação.

4.3 Dspace

O sistema Dspace (http://www.dspace.org/) (SMITH et al, 2003) foi criado para possibilitar o desenvolvimento de repositórios digitais com funções de captura, distribuição e preservação da produção intelectual, permitindo sua adoção por qualquer instituição em forma consorciada federada. Os repositórios DSpace permitem o gerenciamento da produção científica em qualquer tipo de conteúdo digital, dando-lhe maior visibilidade e garantindo a sua acessibilidade ao longo do tempo.

De forma geral o DSpace permite realizar a catalogação de todos os metadados de um OA quando esses metadados seguem um padrão relativamente simples, com poucos metadados, como é o caso do Dublin Core não qualificado (RFC 5013), formado por 8 tipos de metadados distintos.

O DSpace pode ser adaptado para o uso com outros tipos de metadados mais complexo. A recente adaptação do DSpace para operar com os metadados OBAA (ferramenta DSpace-OBAA disponível em www.portalobaa.org), resultou em um aplicativo complexo por ter uma grande quantidade de informações a serem preenchidas de forma manual. Nesse caso o uso do Dspace torna-se difícil e trabalhoso (ver Figura 12), por conta dos mais de 130 metadados distintos definidos pelo padrão OBAA. Isso exige o preenchimento de um grande volume de informações, podendo ocasionar uma eventual desistência dos desenvolvedores ou projetistas do OA de efetuar a catalogação completa e correta deste objeto.

Na figura 16 é mostrada a ferramenta de catalogação DSpace-OBAA com os onze grupos de metadados do OBAA (GLUZ e VICARI, 2011), em destaque as etapas que o desenvolvedor de Objeto de Aprendizagem deve percorrer para preencher todas as informações do catalogo para que os metadados do objeto de aprendizagem estejam corretos.

Figura 17 – Interface do DSPACE-OBAA

Fonte autor (2012).

4.4 Análise Comparativa

Após a utilização das ferramentas citadas acima, foi empreendida uma análise comparativa entre as ferramentas para verificar suas funcionalidades, aspectos positivos e negativos em relação a catalogação de metadados.

Para as ferramentas eXe Learning e Xerte a autoria de alguns objetos de aprendizagem foi realizada com o intuito de avaliar as ferramentas nos itens: tempo gasto para completar o processo de autoria, a utilização de inteligência das ferramentas no auxílio ao usuário para completar a tarefa, sua interface, a cobertura das ferramentas em função do tipo de OA e a utilização de ontologias no processo. Nos experimentos realizados nas ferramentas de autoria de conteúdo foram criados OAs para o domínio de matemática para o ensino médio, com as seguintes características: inclusão de uma imagem, a inclusão de uma tabela, a inclusão uma apresentação e a inclusão de um apêndice de um livro. Logo após a autoria OA a catalogação de seu metadados foi executada. Para estes processos de autoria de OA e a catalogação de um conjunto de metadados básicos de utilização genérica nas ferramentas de autoria, espera-se que o processo seja executado com o gasto de pouco tempo para o término de tarefa.

Para a ferramenta de catalogação de metadados DSpace o experimento executado para avalição e análise de resultados decorreu pelo processo de catalogação dos OA criados nos experimentos anteriores nas ferramentas de autoria, em que foram catalogados na base do DSpace. Também, espera-se que este processo seja executado do seu inicio ao término da operação em um curto período de tempo e que a ferramenta forneça algum tipo de suporte ao seu preenchimento.

Um item que chama a atenção em todas as ferramentas é o dispêndio de tempo para a conclusão de todas as tarefas necessárias para a finalização da autoria nos aplicativos eXe Learning e Xerte, e para o preenchimento das informações no DSpace. Todas as ferramentas demandam uma quantidade de tempo elevado para a conclusão total de suas tarefas por parte do usuário, levando em conta nos dois primeiros que a autoria do OA é a principal componente de tempo.

Com o auxílio de um usuário com pouco conhecimento técnico de OA, foi executado um teste de operação nas ferramentas em relação ao auxílio da ferramenta para que se obtivesse a conclusão da tarefa de autoria ou catalogação dos OAs. Analisando e observando o item referente a inteligência da ferramenta quanto a utilização de recursos voltados a capacidade de dedução de informações para o preenchimento de metadados, como agentes inteligentes, *wizards* ou oferecem a possibilidade de preenchimento dos metadados mais comuns nos objetos de aprendizagem com informações retidas da autoria do conteúdo. A utilização de metodologias como agentes inteligentes e *wizards* possibilita a otimização do processo de catalogação, melhorando a qualidade da informação, por sua vez facilitando a busca, o compartilhamento e a reutilização do Objeto de Aprendizagem por outros.

As ferramentas não possuem suporte e também não dispõem de recursos de inteligência para a realização de suas tarefas e auxílio ao projetista de Objeto de Aprendizagem, o eXe Learning e o Xerte utilizam *Dialog Box* agrupados por nível de afinidade e dicas do tipo *tooltip* para orientação do usuário. Já o DSpace não possui nenhum recurso para auxílio ou orientação do projetista de Objeto de Aprendizagem.

O conhecimento técnico do usuário em relação aos metadados dos OAs deve estar concreto em relação ao número de metadados, a identificação destes metadados, qual o tipo destes dados, como estão classificados, etc. Estas informações técnicas são de grande importância para usuário dos sistemas analisados.

A interface da ferramenta é outro ponto analisado, a verificação deste item se faz necessária por muitas vezes os projetistas de OA não disporem de conhecimento técnico suficiente sobre os metadados do OA que esta em processo de autoria e catalogação. Desta forma a ferramenta deve possuir uma interface amigável de fácil compreensão, sem a

necessidade de o usuário ter que buscar recursos externos a ferramenta a ponto de desistir de utilização.

Das ferramentas analisadas o DSpace possui uma interface pouco amigável para usuários sem conhecimento técnico em sobre os metadados do OA. Por outro lado, as interfaces das outras ferramentas são mais amigáveis possibilitando a edição dos conteúdos dos Objetos de Aprendizagem de forma relativamente fácil.

O próximo critério está relacionado com a cobertura da ferramenta em relação aos metadados de Objeto de Aprendizagem de um determinado padrão: se completo, caso a ferramenta permita a edição de todos os metadados deste padrão, ou se limitado, quando a ferramenta permite apenas a edição de alguns metadados.

O item final a ser analisado diz respeito a representação de metadados através de ontologias. A utilização deste recurso permite a inferência sobre domínios específicos representados pelas ontologias, tornado desta forma a possibilidade de oferecer inteligência ao processo.

Para o item representação de metadados através de ontologias, nenhuma das ferramentas analisadas possui suporte ou algum tipo de tecnologia para oferecer inteligência ao processo de catalogação dos objetos de aprendizagem gerados por estas ferramentas.

Tabela 2 – Análise de ferramentas

Ferramenta / Itens analisados	Tempo para completar a operação	Inteligência	Interface de fácil operação e compreensão	Cobertura	Uso de Ontologias
DSpace	Variável de acordo com a quantidade de metadados a serem preenchidos	Não	Não	Completa (DCMI)	Não
eXe Learning	45 a 50 min	Não	Sim	Limitada (DCMI)	Não
Xerte	45 a 50 min	Não	Sim	Limitada (SCORM)	Não

Fonte: autor (2012).

Na autoria de Objeto de Aprendizagem as ferramentas eXe Learning e Xerte se destacam em recursos e praticidade, o DSpace não possui suporte a autoria de objetos, centrando-se nas informações do objeto a ser catalogado. O DSpace solicita uma grande quantidade de informações do usuário durante a catalogação do objeto. As outras ferramentas Xerte e eXe Learning suportam apenas metadados básicos para criação de objetos de aprendizagem criando uma certa dificuldade quando se tem a intensão de reutilizar conteúdos, a localização deste objetos por terceiros pode ser prejudicada com a limitação de metadados do objeto.

A expectativa do sistema *Linnaeus* é reduzir de forma significativa o volume de informações que o projetista tem que catalogar. As ferramentas vistas nessa seção oferecem um tratamento limitado para a questão de catalogo de metadados para Objetos de Aprendizagem, em que nas duas primeiras ferramentas listadas é utilizado apenas metados genéricos para abranger um grande número de OA e na terceira ferramenta analisada a grande quantidade de informações a serem preenchidas. No *Linnaeus*, por outro lado, espera-se reduzir o trabalho de projetistas de OA através de um extensivo uso de *wizards* de apoio a catalogação que incorporam os conhecimentos de ontologias sobre domínios de ensino e

sobre aplicações educacionais (incluindo interoperação e acessibilidade) que sejam compatíveis com a ontologia de metadados OBAA (GLUZ e VICARI, 2011).

5 LINNAEUS SISTEMA PARA CATALOGAÇÃO DE OA

Este capítulo apresenta o Sistema *Linnaeus* que foi desenvolvido como uma ferramenta inteligente de apoio a edição de metadados de OA. O *Linnaeus* utilizará as tecnologias de agentes inteligentes, na forma de *wizards*, e de ontologias educacionais para proporcionar um processo prático e efetivo para a catalogação de OA.

O nome *LINNAEUS* é inspirado em *Carl Linneaus* criador da taxonomia moderna e catalogação (KOERNER, 1999). Tendo essa inspiração em mente, espera-se que a principal característica deste sistema devera ser o auxílio ao projetista (*designer*) ou desenvolvedor de objetos de aprendizagem a catalogar de forma rápida e fácil os novos objetos, sem a necessidade do conhecimento técnico sobre os metadados do OA por parte do projetista. O sistema *Linnaeus* será encarregado de editar os metadados dos objetos de aprendizagem suportados pela plataforma MILOS (GLUZ, VICARI e PASSERINO, 2012).

A seguir são apresentadas a arquitetura onde o sistema *Linnaeus* esta inserido a plataforma MILOS (GLUZ, VICARI e PASSERINO, 2012) e as principais características, a visão geral do sistema *Linnaeus*, juntamente com a descrição da arquitetura de *software* do Sistema *Linnaeus*, seus casos de uso e a apresentação de seu mecanismo de inferência.

5.1 Infraestrutura MILOS

A infraestrutura MILOS (*Multiagent Infrastructure for Learning Object Support*) fornecerá suporte aos processos de autoria, gerência, busca e disponibilização de Objetos de Aprendizagem compatíveis com a proposta de padrão de metadados OBAA (GLUZ e VICCARI, 2011). O objetivo a longo prazo do projeto da infra-estrutura MILOS é especificar e implementar uma arquitetura de agentes para a MILOS que seja capaz de suportar requisitos de adaptabilidade, interoperabilidade e acessibilidade previstos pela proposta OBAA.

Para atingir seus objetivos, os metadados OBAA englobam um conjunto extenso de informações de catálogo para o conhecimento e interação de um desenvolvedor. Esta situação se evidencia quando verificam-se os processos relacionados ao ciclo de vida dos OA: autoria dos próprios conteúdos, a catalogação destes conteúdos (essencialmente autoria de metadados), o armazenado e o gerenciado dentro de um repositório de um OA.

A distribuição destas tarefas entre especialistas distintos, visando prover pessoal capaz de tratar de questões tecnológicas relacionadas aos OA, apesar de possível, pode não ser prática quando são adicionados os custos econômicos extras de contratação deste pessoal com conhecimento tecnológico ao contexto de um processo educacional que já comporta um grande número de agentes (GLUZ, VICARI e PASSERINO, 2012).

Desta forma, para se tornar possível a utilização eficaz de Objetos de Aprendizagem é necessário que haja um suporte tecnológico adequado. A infra-estrutura MILOS propõe a utilização de componentes de *software* especializados na execução das diversas tarefas relacionadas ao ciclo de vida dos Objetos de Aprendizagem para que os desenvolvedores de conteúdo educacionais possam abstrair conhecimentos técnicos e tecnológicos sobre padrões de Objetos de Aprendizagem, metadados, etc. Em particular, espera-se que usuários sem um *background* tecnológico sejam capazes de desempenhar as seguintes tarefas com o apoio dos *softwares* e serviços da MILOS (GLUZ, VICARI e PASSERINO, 2012).

- Gerar conteúdos e catalogar (especificar) seus metadados de forma a suportar requisitos de acessibilidade, de educação, multimídia e multiplaforma;
- Adaptar e distribuir os Objetos de Aprendizagem diretamente para os diferentes tipos de plataformas tecnológicas suportadas pelo OBAA (Web, TVD e dispositivos móveis, em um primeiro momento);
- Localizar e disseminar (publicar) os Objetos de Aprendizagem;
- Utilizar Objetos de Aprendizagem;
- Armazenar e gerenciar repositórios de Objetos de Aprendizagem.

O projeto OBAA teve seu término com sucesso, tendo sua proposta de metadados para um Objeto de Aprendizagem multimídia e multiplataforma sido aceita preliminarmente pelo MEC, constituindo uma base sólida para um futuro padrão nacional para Objetos de Aprendizagem. Entretanto, para que isto se torne realidade, são necessárias várias ações adicionais. Dentre elas está à infraestrutura MILOS. Para criação de componentes de *software* especializados que possam apoiar as tarefas citadas, o projeto da infra-estrutura MILOS se baseia principalmente a Engenharia de Ontologias e a Engenharia de *Software* Orientada a Agentes.

Ontologias permitem especificar de forma rigorosa e padronizada as propriedades de um domínio de aplicação, mas não oferecem elementos ativos do sistema que desenvolva esta aplicação. Já a Engenharia de *Software* Orientada a Agentes possibilita projetar e desenvolver sistemas baseados em agentes de *software* capazes de desenvolver aplicações cujas propriedades foram especificadas por meio de ontologias (GLUZ, VICARI e PASSERINO, 2012).

Os agentes de *software* encapsulam o conhecimento que seria inviável, ou muito custoso de um ser humano aprender. Os agentes propostos pelo projeto abstraem o conhecimento dos padrões de metadados e especificidades tecnológicas dos desenvolvedores de Objeto de Aprendizagem permitindo a estes direcionem seu foco de trabalho em seu objetivo que é prover conteúdo útil.

Os agentes propostos pelo projeto são agrupados em subsistemas apresentados na figura seguinte:

- Sistema de Busca: desenvolve o suporte as atividades de localização dos Objetos de Aprendizagem federações de catálogos de Objeto de Aprendizagem.
- Sistema de Apoio Pedagógico: desenvolve o suporte as atividades de uso dos Objeto de Aprendizagem.
- Sistema de Autoria: desenvolve o suporte as atividades de autoria de Objeto de Aprendizagem, incluindo suporte a adaptação multiplataforma.
- Sistema de Gerência: suporte as atividades de armazenamento, gerenciamento, publicação/distribuição multiplataforma de Objetos de Aprendizagem.

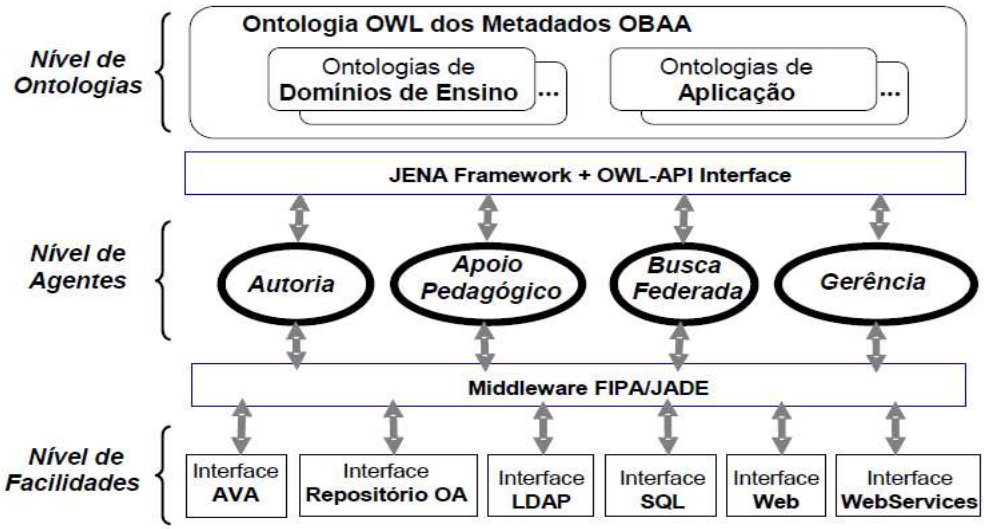
Os agentes de software na infraestrutura MILOS devem desenvolver as funcionalidades e serviços que a infraestrutura deve disponibilizar aos seus diversos usuários. Faz-se necessário a diferenciação da atuação dos agentes do papel das ontologias: embora as ontologias possam especificar as propriedades de um domínio de aplicação, definindo inclusive as ações, percepções, algoritmos e heurísticas que são possíveis neste domínio, elas são essencialmente especificações estáticas deste conhecimento. Desta forma são os agentes

de *software* que realmente utilizam o conhecimento estático em comportamento dinâmico para tentar prover auxílio aos usuários.

A camada de serviços oferece facilidades de interface para *Web* permitindo que os agentes e sistemas multiagente da MILOS possam ser acessados por ambientes e aplicações educacionais, além de permitir também que estes agentes tenham acesso aos repositórios de Objetos de Aprendizagem, serviços de diretórios, plataformas de EAD, bancos de dados e servidores *Web* (VICARI e GLUZ, 2011).

A arquitetura de *software* definida para a infraestrutura de agentes MILOS está dividida em três grandes níveis de abstração como pode ser vista na Figura 17:

Figura 18 − Organização Geral da Infraestrutura MILOS



Fonte: VICARI e GLUZ (2011).

5.2 Visão Geral do Sistema

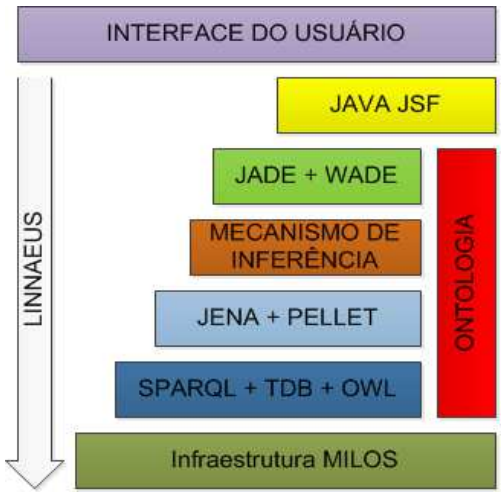
O objetivo do sistema *Linnaeus* é fornecer uma ajuda inteligente e semi-automatizada ao processo de catalogação de OA, permitindo a geração de parte dos valores dos metadados suportados pela proposta OBAA[1] (GLUZ, VICARI e PASSERINO, 2012). O processo de catalogação leva em consideração ontologias de domínios educacionais simultaneamente com a ontologia de metadados OBAA . Neste processo de catalogação é possível uma maior cobertura do conhecimento sobre o domínio educacional do objeto sendo criado, possibilitando o preenchimento dos metadados com maior precisão e exatidão nas informações.

O protótipo deste sistema foi desenvolvido utilizando à linguagem de programação JAVA com suporte a *web*, atendendo umas premissas do projeto OBAA-MILOS de que suas ferramentas devem ser *open source* de domínio público. A construção do sistema *Linnaeus* se

[1] Segunda versão da ontologia desenvolvida para atender as premissas do projeto OBAA-MILOS(GLUZ, VICARI e PASSERINO, 2012), disponível em: <http://obaa.unisinos.br/obaa22.owl>.

tornou possível pela utilização combinada de diversas ferramentas tecnológicas. Na figura 19 são mostradas as principais ferramentas tecnológicas utilizadas na construção do sistema. Essa combinação de ferramentas propiciam suporte a *web* semântica, permitem o desenvolvimento de sistemas multiagentes, além de implementar mecanismos de raciocínio lógico dos agentes.

Figura 19 - Tecnologias utilizadas no Sistema *Linnaeus*



Fonte autor (2013).

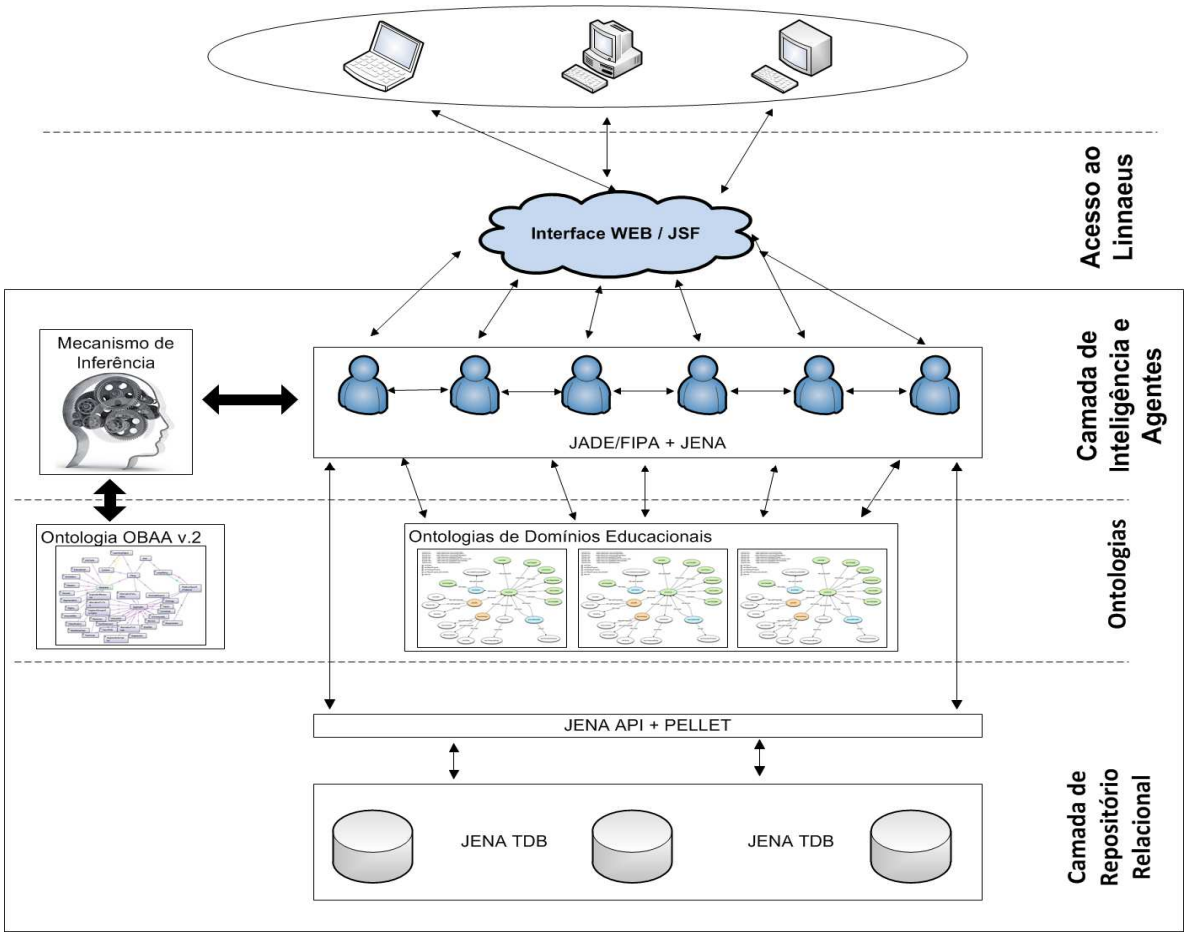
As tecnologias utilizadas na elaboração do sistema *Linnaeus* (figura 19) incluem: a linguagem de programação JAVA utilizando o *framework* JSF (Java Server Faces) para construção de uma interface de usuário baseada em componente para aplicações *web*, a plataforma JADE/FIPA como ambiente multiagente, o sistema de inferência *JRules* integrado ao JAVA, o *framework* JENA juntamente com a biblioteca de inferências PELLET para o acesso e validação de ontologias.

5.3 Arquitetura Geral do Sistema Linnaeus

Seguindo as características gerais da Infraestrutura MILOS (ver seção 5.1) o sistema *Linnaeus* foi organizado em quatro camadas distintas (ver Figura 20):

- Camada de Acesso: nível responsável pela interação do *Linnaeus* com o usuário, através de páginas *web*, e também pela interação entre outros sistemas através de *Web Services*.
- Camada de Agentes: nível formado por agentes inteligentes de *software* (*wizards*) que usam mecanismos de raciocínio lógico para inferir os valores dos metadados do OA sendo criado pelo usuário.
- Camada de Ontologias: este nível é formado por exclusivamente por ontologias, utilizadas no mecanismo de inferência e também acessado por alguns agentes do sistema para alimentar a base de fatos do mecanismo de inferência.

Figura 20 - Arquitetura do Sistema Linnaeus



Fonte autor (2013).

- Camada de Armazenamento: este nível do sistema é constituído por ferramenta de armazenamento semântico JENA TDB, tendo suporte para OWL nativo e os metadados OBAA estarão armazenados. Este sistema utiliza os serviços de armazenamento da MILOS, implementados no sistema MSSearch (DA SILVA, 2013).

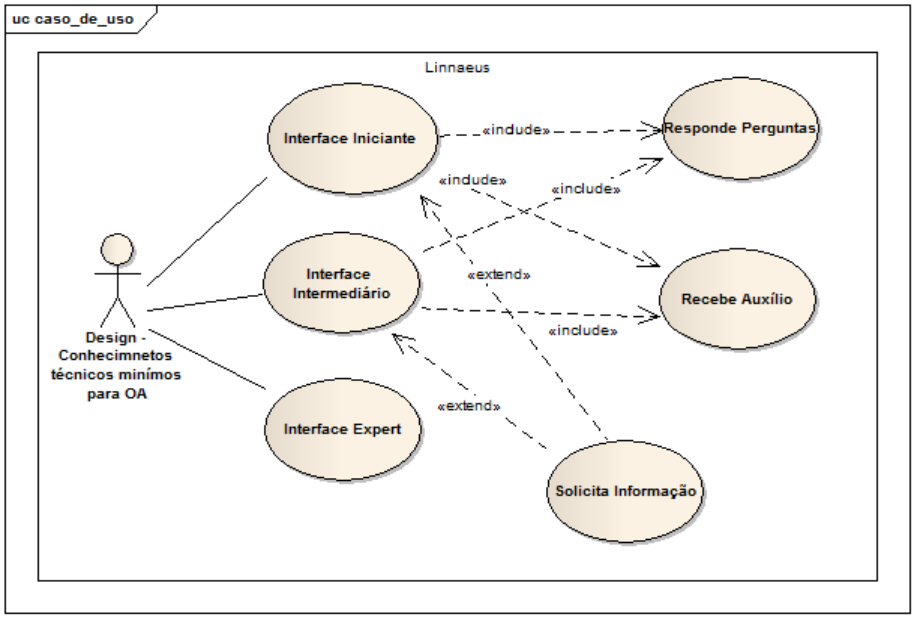
As operações e tarefas executadas pelo *Linnaeus* são implementadas por meio de agente de *software* (ver seção 3), usando o conceito de agentes *wizards* inteligentes. A criação do catálogo de metadados do sistema se dará de forma automática sem o conhecimento do usuário, o catálogo seguirá como uma de suas premissas o domínio do OA, sua aplicabilidade, plataforma de operação, para a criação do catálogo.

5.4 Casos de Uso do Sistema

O protótipo do sistema *Linnaeus* foi projetado para ser utilizado por desenvolvedores de OA. O *Linnaeus* oferece distintos de interface de usuário de acordo com os conhecimentos do desenvolvedor em relação aos aspectos técnicos dos OA. São considerados três níveis de conhecimentos: desenvolvedores leigos, sem conhecimentos técnicos sobre OA ou

metadados, desenvolvedores com conhecimentos intermediários sobre estes temas e desenvolvedores que se consideram especialistas (*experts*) nestes conhecimentos. Seguindo esta caracterização, as interfaces de usuário do *Linnaeus* são especificadas em três casos de uso distintos (ver Figura 21), sendo o primeiro para usuários classificados como iniciantes devido ao nível de conhecimento técnico sobre OA, o segundo em nível intermediário de conhecimento técnico e o terceiro em nível expert com grande conhecimento técnico em OA.

Figura 21 – Caso de uso Sistema Linnaeus

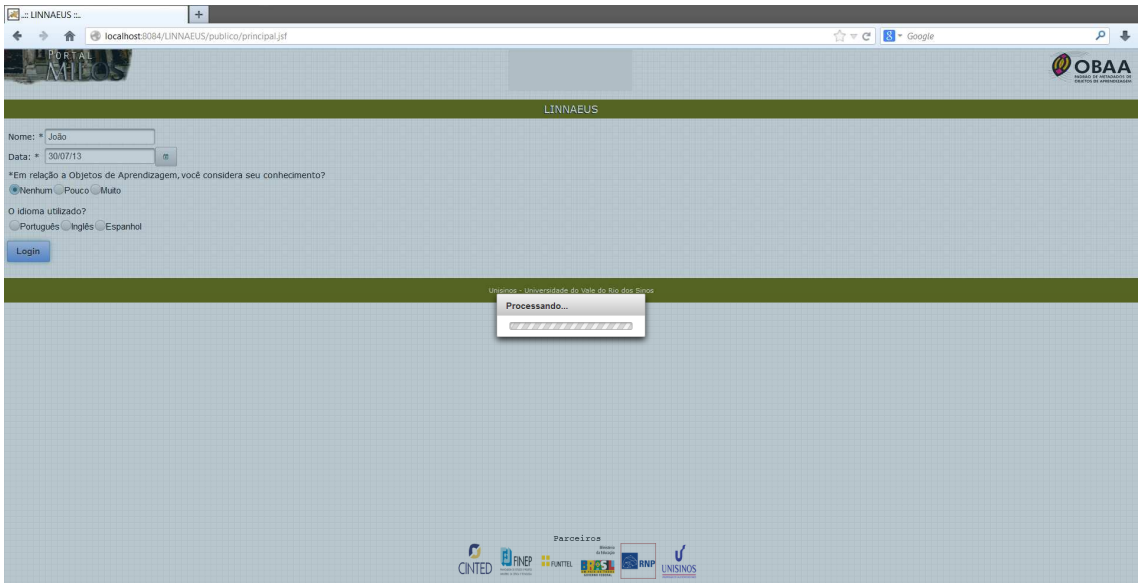


Fonte autor (2013).

A seguir são detalhados as principais características destes casos de uso. No primeiro caso de uso considera-se que o desenvolvedor tenha escolhido a opção de conhecimentos básicos em OA (leigo), já no segundo caso apresentado o considerou-se que o desenvolvedor de OA possua conhecimentos avançados em OA (*expert*). O sistema possui sua interação com o usuário através de *pop-ups web* solicitando informações ao usuário sempre que o mecanismo de inferência necessitar de informações adicionais ao processo. As perguntas são geradas pelo agente que coordena o mecanismo de inferência, estas perguntas são geradas de uma base de perguntas pré-selecionadas para que o usuário possa respondê-las de forma imediata sem a necessidade de buscar informações e ajuda para tal questionamento. O *Linnaeus* esta definido com dois tipos de perguntas que serão aplicadas ao desenvolvedor de OA, a primeira pergunta é do tipo *EditBox*, para entrada de dados do tipo texto. O segundo tipo de pergunta é do tipo *RadioBox* para respostas do tipo FALSO/VERDADEIRO ou SIM/NÃO.

Quando o sistema é acessado, o usuário encontra a área onde deve fornecer informações de identificação (figura 22), seu grau de conhecimento técnico com relação aos OA, juntamente com o idioma nativo do OA sendo catalogado. No momento em que o *login* é executado os agentes do sistema juntamente com o mecanismo de inferência são inicializados e o processo de catalogação de metadados fica em modo de espera para realizar a consulta na base de ontologias para o domínio definido pelo desenvolvedor de OA.

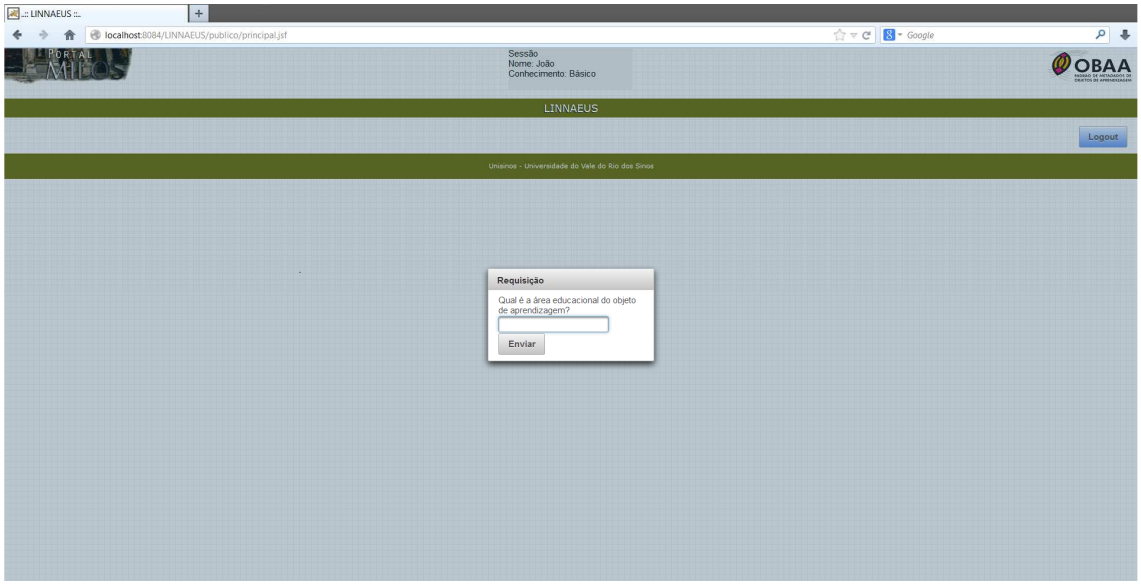
Figura 22 - Tela de *Login* do Sistema Linnaeus



Fonte autor (2013).

Na próxima interação do sistema é solicitado ao desenvolvedor informar o domínio educacional do OA que será catalogado. Através de um entrada de dados do tipo *EditBox* (figura 23) com esta informação o sistema executa a pesquisa na base ontologias de domínios educacionais e transmite estas informações ao agente que implementa o mecanismo de inferência para a continuidade do processo de preenchimento de metadados. Quando este agente não conseguir resolver qual informação será atribuída a determinado metadado, então é enviada uma solicitação de informações adicionais ao agente de gerenciamento da catalogação. Este agente, por sua vez, deve encaminhar a solicitação para o desenvolvedor de OA, enviando, posteriormente, a resposta do usuários para o agente de inferência. Tal processo deve se repetir até o termino do preenchimento dos metadados.

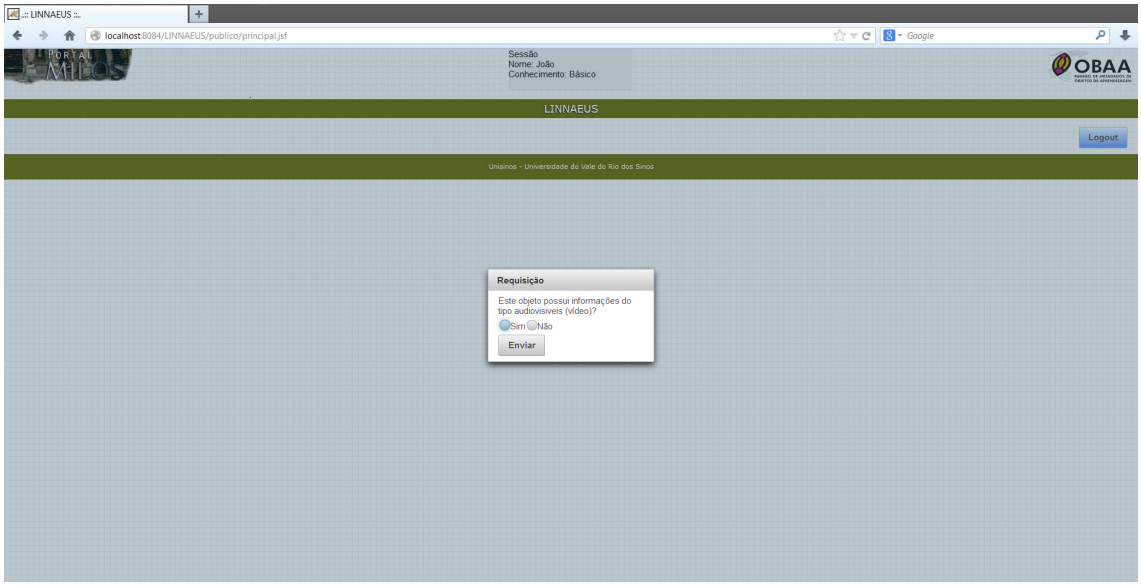
Figura 23 - Tela de entrada de dados do tipo texto fornecidos pelo desenvolvedor de OA



Fonte autor (2013).

Por exemplo, na eventual necessidade de informações do tipo SIM/NÃO por parte do desenvolvedor de OA, então é elaborada uma pergunta utilizando o formato *RadioBox* para receber a resposta do desenvolvedor. Na figura 24 é mostrado a solicitação do sistema ao desenvolvedor de OA utilizando um questionamento do tipo *RadioBox*.

Figura 24 - Tela de entrada de dados do tipo FALSO/VERDADEIRO fornecido pelo desenvolvedor de OA

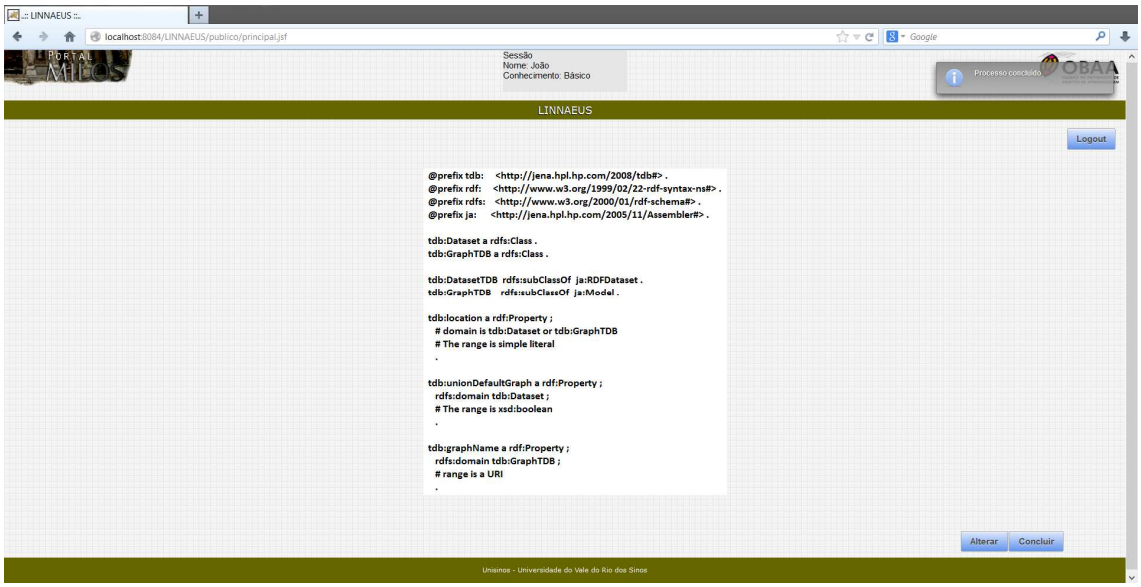


Fonte autor (2013).

O sistema *Linnaeus* interage com o usuário a quantidade de vezes necessária até encerrar o preenchimento dos metadados do OA que esta sendo catalogado. Se o sistema não encontrar nenhum problema ou dificuldade para o preenchimento dos metadados, e o processo tiver percorrido todos os itens definido pelo padrão de metadados OBAA (GLUZ, VICARI e PASSERINO, 2012) o sistema finaliza todas as tarefas e gera os metadados para serem gravados na base de dados RDF.

O próximo passo a ser executado pelo desenvolvedor de OA é a finalização o processo. O sistema apresenta ao usuário os metadados gerados (figura 25) que podem ser alterados por edição manual sem o auxílio do sistema de agentes ou o desenvolvedor poderá finalizar o processo aceitando os metadados gerados pelo sistema através do botão de conclusão da tarefa. Ao executar o botão de conclusão o sistema dispara a gravação dos metadados na base de dados RDF e uma nova sessão é iniciada para a criação de um novo catalogo.

Figura 25 - Tela final de apresentação dos metados gerados pelo sistema no formato de triplas RDF



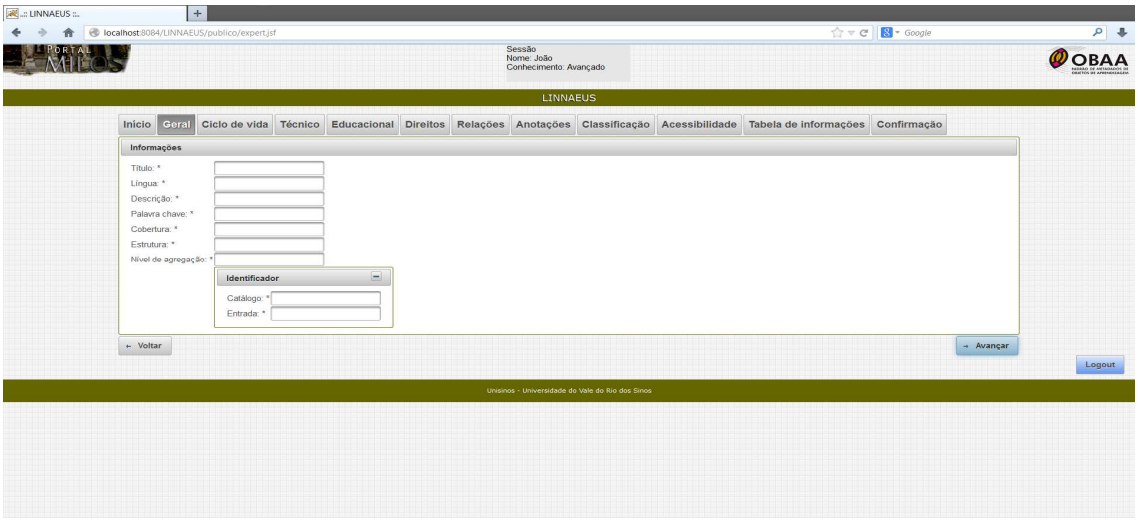
Fonte autor (2013).

O próximo caso de uso apresentado tem suas funcionalidades direcionadas a estrutura dos metadados OBAA (GLUZ, VICARI e PASSERINO, 2012) na sua forma original, ou seja, os metadados são solicitados ao usuário em sua forma mais bruta conforme a especificação do formato de metadados OBAA.

Quando o desenvolvedor de OA tiver conhecimentos avançados das características técnicas dos objetos de aprendizagem, poderá escolher a opção de perfil avançado do sistema. Ao selecionar esta opção o sistema direciona o desenvolvedor de OA para o ambiente em que todos os itens dos metadados deveram ser percorridos para seu preenchimento até a chegada ao último item para a finalização do processo. Nesta opção o desenvolvedor não possui suporte algum do sistema de inferência ou de algum agente para o preenchimento dos metadados. O preenchimento dos metadados na sessão para perfil avançado segue a sequência definida pelo padrão OBAA de metadados (1. *General*, 2. *Lyfe Cycle*, 3. *Meta-Metadata*,etc...).

Na figura 26 é mostrada a primeira solicitação de informações (*General*) para desenvolvedores de OA em nível *expert* de conhecimento em relação a características técnicas dos OAs. É possível notar a organização e distribuição dos grandes grupos de metadados OBAA que devem ser percorridos pelo desenvolvedor para conclusão da tarefa de preenchimento manual dos metadados.

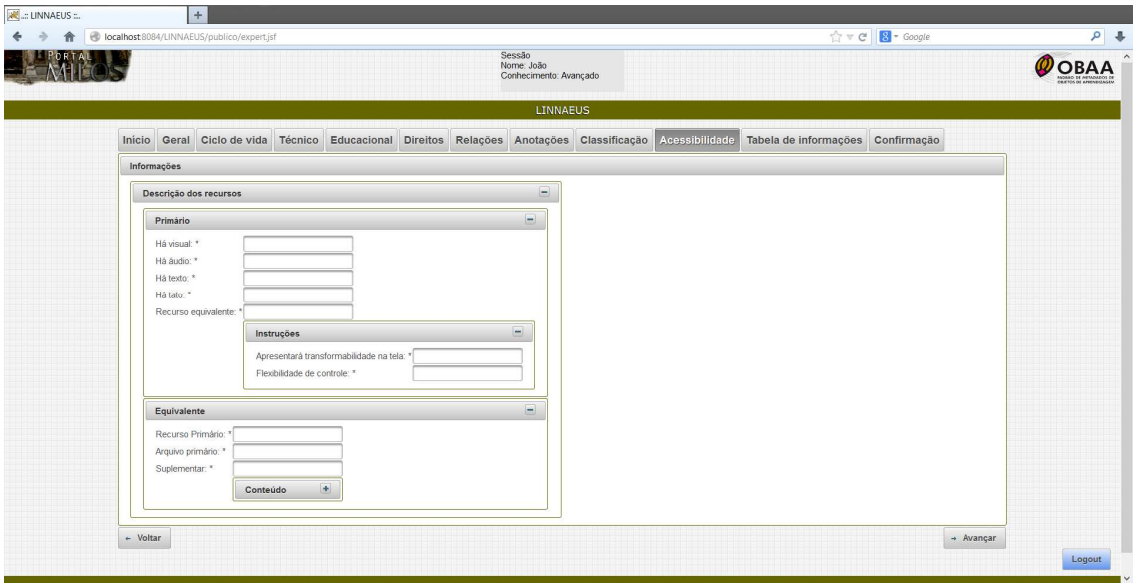
Figura 26 - Primeira tela de informações a serem preenchidas pelo desenvolvedor *expert*



Fonte autor (2013).

A sessão para desenvolvedores de OA com conhecimentos avançados sobre as características técnicas dos objetos de aprendizagem foi desenvolvida visando comportar a todos os itens de metadados suportados pelo OBAA, nesta sessão os metadados são distribuídos em dez (10) (vide figura 27) grandes grupos listados na parte superior do *Linnaeus*. E em cada um destes grupos estão distribuídos todos os metadados que devem ser preenchidos pelo desenvolvedor, os itens que tem seu preenchimento obrigatório dentro de cada grupo possui uma marca com o símbolo asterisco (*) para sua identificação, o próprio *Linnaeus* não permite que o desenvolvedor prossiga para o próximo grupo sem o preenchimento correto dos metadados assinalados como obrigatórios dentro do grupo.

Figura 27 - Tela para preenchimento dos metadados, para usuário *expert*



Fonte autor (2013).

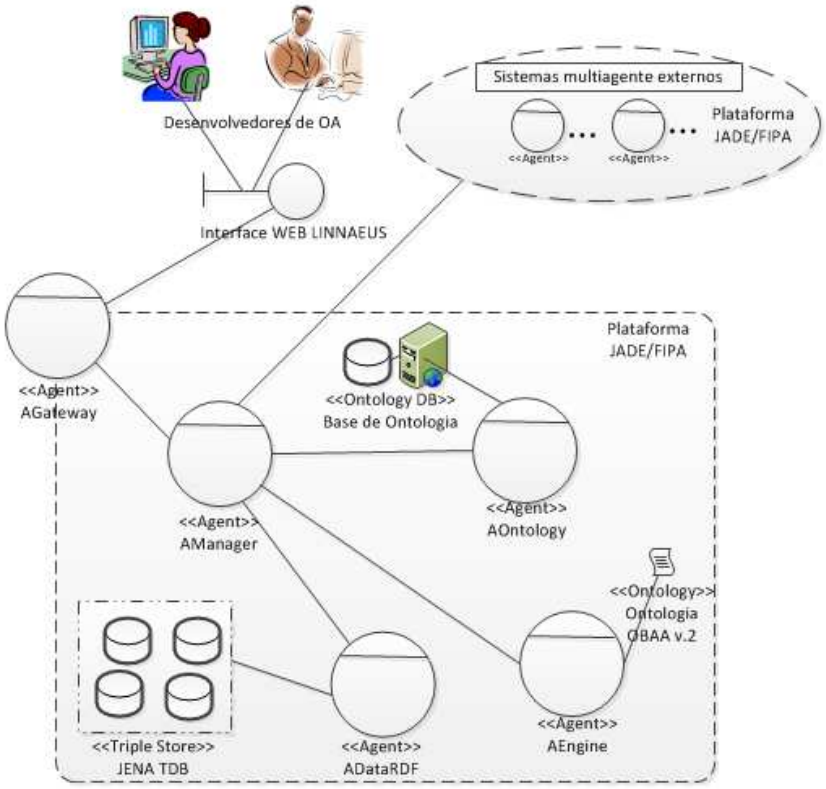
Para o caso de uso relacionado aos desenvolvedores de nível de conhecimento intermediário sobre OA, o sistema também utiliza recursos de agentes de *software* operando

como *wizards*, eventualmente solicitando determinadas informações do desenvolvedor. Na interface para usuários intermediários são utilizados os mesmos recursos e características descritas para a interface de usuários leigos. A diferença é que o *Linnaeus* realiza uma quantidade maior de solicitações e interrupções do desenvolvedor, diferente da interface para usuários leigos onde o sistema tenta preencher o maior número possível de informações sobre os metadados OBAA, interrompendo o mínimo possível o desenvolvedor de OA.

5.5 Agentes do sistema

O sistema *Linnaeus* é constituído por cinco agentes que têm papéis distintos dentro da arquitetura. Uma visão geral da disposição dos agentes e sua relação com o sistema é apresentada na figura 28. Os agentes do *Linnaeus* tem seu comportamento modelado de acordo com a ferramenta Jade/WADE (BERGANTI, et al. 2012). Essa ferramenta permite não apenas modelar o comportamento de um agente, mas também é capaz de gerar o código fonte inicial, em JAVA, deste agente.

Figura 28 – Disposição dos agentes na arquitetura do sistema Linnaeus



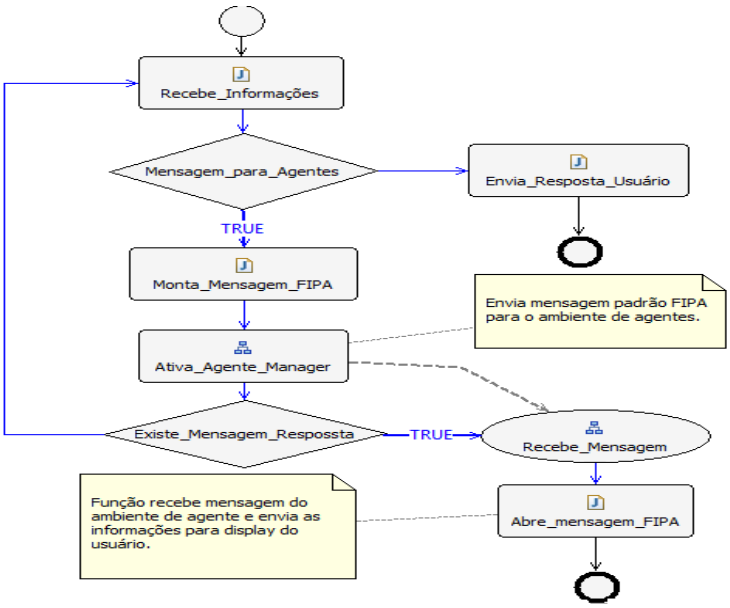
Fonte autor (2013).

5.5.1 Agente AGateway

O agente *AGateway* tem seu comportamento (vide figura 29) baseado em um *gateway* para troca de informações entre o ambiente de agentes e a interface de usuário, em ambas estão em meio de trabalho distintos. Este agente é responsável por receber as solicitações e

respostas do usuário e converter estas informações em uma mensagem padrão FIPA para a transmissão desta ao ambiente de agentes. E a conversão da resposta ou requisição de informações provenientes do sistema de agentes ao sistema JSF gerenciador da interface de usuário, convertendo também a mensagem FIPA recebida em um formato padrão para classes JSF.

Figura 29 - Modelagem comportamento Agente *AGateway* com JADE/WADE.



Fonte autor (2013).

O comportamento do agente *AGateway* modelado com a ferramenta JADE/WADE é apresentado na figura 29. Este agente inicia suas operações ao receber informações ou requisições da interface de usuário. Neste processo o agente gera a mensagem FIPA para ser repassada ao ambiente de agentes, mais especificamente o agente *gateway* envia e recebe suas mensagens do agente *AManager* que é o responsável pelo gerenciamento dos demais agentes. Assim que o agente *AGateway* envia a mensagem, este finaliza sua operação para que possa receber a resposta proveniente do ambiente de agentes, para então formatá-la e entregar as informações ao sistema JSF que gerencia o ambiente de usuário, ou receber novas requisições do ambiente de usuário.

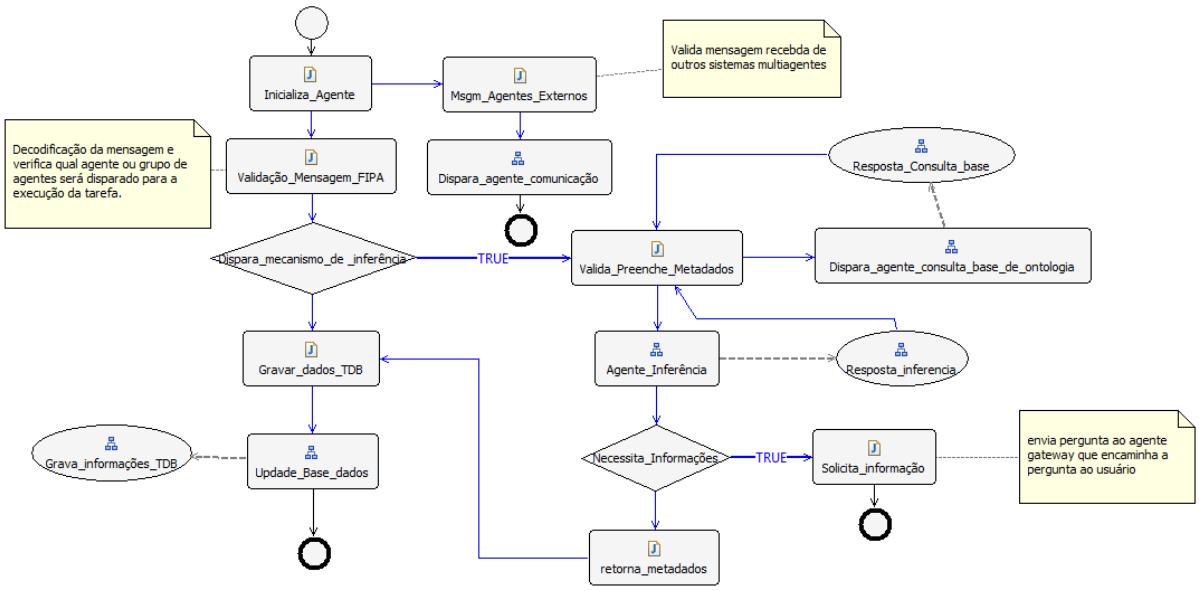
O agente *AGateway* possui em sua memória a formatação para grande parte dos tipos de atos comunicativos existentes no padrão proposto pela JADE/FIPA (vide capítulo 3, sessão 3.1.2). Ao receber da interface de usuário, uma mensagem destinada ao sistema de agentes do *Linnaeus* o agente *AGateway* verifica se a mensagem possui o objeto mensagem() com suas classes *string message*, *string receiver* e *integer acltype* devidamente preenchidos para que este agente possa monta a mensagem padrão FIPA para repassar tal mensagem ao ambiente de agentes. O agente *gateway* é basicamente encarregado de proporcionar a integração entre a interface *web* para os usuários e o ambiente do sistema multiagentes.

5.5.2 Agente AManager

Dentro do sistema multiagentes do *Linnaeus*, o agente encarregado em receber e organizar toda a operação dos demais agentes do sistema é o agente *AManager* (figura 30)

que receber a mensagem proveniente do agente *gateway* e executa a operação necessária para o atendimento da requisição, ou dispara novas requisições ao demais agentes para que em conjunto com estes agentes possa executar sua tarefa com velocidade e precisão

Figura 30 – Modelagem comportamento Agente *AManager* com JADE/WADE.



Fonte autor (2013).

O agente *AManager* tem seu comportamento iniciado pelo recebimento de uma mensagem proveniente do agente *AGateway* que intermedia a comunicação entre o ambiente de agentes e a interface de usuários. Esta mensagem informa ao manager qual a operação este deve realizar, ou quais ações o agente deve executar para o atendimento da requisição do usuário no sistema. Este agente é responsável pela organização e gerenciamento dos demais agentes para que haja um sincronismo entre eles para a correta execução de suas tarefas.

O agente *AManager* com seu comportamento mostrado na figura 30, valida a mensagem recebida, e a mensagem estando consistente o agente verifica no conteúdo da mensagem a ação que deve executar ou o recurso que deve ser disparado para auxiliá-lo na efetivação do processo. Se na verificação da mensagem o agente perceber que a operação a ser executada é a de preenchimento dos metadados por *wizards*, este agente dispara o processo que controla o mecanismo de inferências. Neste processo o agente manager com o auxílio dos agentes responsáveis pela consulta a base ontologias de domínios educacionais e também do agente responsável pelo controle do mecanismo de inferências, executam o processo de preenchimento do maior número de metadados possíveis sem a intervenção do desenvolvedor de OA.

No processo de inferências, o agente *AManager* processa os grupos de metadados OBAA preenchidos pelo mecanismo de inferências, verificando se todos os itens foram preenchidos ou parcialmente preenchidos. Neste último caso o agente dispara uma nova requisição para que o agente de consulta a base de ontologias realize uma nova busca de informações e de posse do retorno da consulta o agente manager solicita ao agente responsável pelo mecanismo de inferências que use estas novas informações para preencher mais campos do grupo de metadados atuais.

No momento em que o agente de inferência não conseguir preencher mais itens dos metadados em questão, o agente executa a solicitação de informações ao usuário através de um determinado tipo de pergunta pré-definida no comportamento do agente *AManager*, e esta pergunta é caracterizada por dois tipos distintos de informação: para as informações do tipo verdadeira/falso ou sim/não o agente monta uma pergunta do tipo *radiobox* com as duas alternativas para que o desenvolvedor de OA possa selecionar uma alternativa, o outro tipo de informações requisitadas pelo agente são as do tipo texto, em que o agente envia uma pergunta do tipo *editbox* ao desenvolvedor de OA, que deve preencher o campo com informações do tipo alfanuméricas e finaliza sua execução. Ao receber a mensagem contendo a resposta fornecida pelo desenvolvedor de OA o agente *manager* retoma o processo de inferências, executando uma nova consulta a base de ontologias e passando estas consultas ao mecanismo de inferência para executar a continuidade do preenchimento das informações dos metadados. Executando o processo de consulta a base de ontologias e consulta ao usuário até percorrer todos os grupos de metadados descritos pelo OBAA.

Ao finalizar o processo de preenchimento dos metadados o agente apresenta as informações inferidas para cada campo de metadado do padrão OBAA, requisitando a confirmação do desenvolvedor de OA para que o processo de gravação na base de dados RDF seja executado pelo agente responsável por gerenciar as informações da base de dados. Na apresentação dos metadados preenchidos o desenvolvedor de OA possui a opção de alterar manualmente os metadados como queira, porém sem o auxílio do sistema de agentes.

No momento em que o *AManager* recebe a mensagem de confirmação do usuário, de que os metadados podem ser gravados é disparada uma mensagem ao agente gestor da base de dados, que recebe a mensagem com todas as informações a serem armazenadas na base de dados RDF, este executa o processo de verificação da consistências dos dados, e logo após executa a gravação na base RDF. O agente manager recebe a mensagem de retorno, informado gravação correta das informações, envia uma mensagem de finalização de sessão ao agente *gateway* que apresenta ao desenvolvedor de OA a mensagem que o processo de catalogação executado por ele finalizou com sucesso e encerra a sessão.

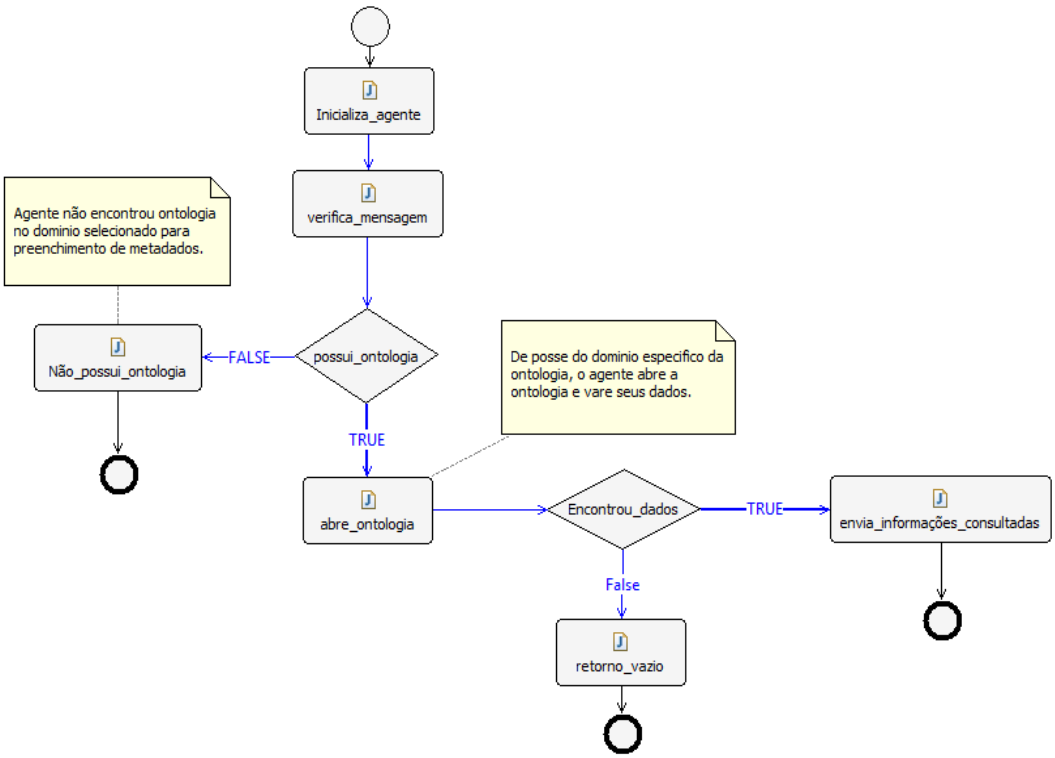
5.5.3 Agente AOntology

As consultas à base de ontologias de domínios educacionais são realizadas pelo agente *AOntology*. Este agente inicia sua operação inicializando suas variáveis internas, e executando uma varredura na biblioteca de ontologias gerando assim um índice de conteúdo contendo os domínios das ontologias presentes em sua base. Após inicializar seus indicadores e variáveis o agente entra em operação normal (ver Figura 31) decodificando mensagens FIPA, identificando o domínio de trabalho, o conteúdo ou instruções de pesquisa e tipo de informações que devem ser retornadas.

De posse de suas tarefas o agente *AOntology* valida o domínio de trabalho com seu índice de conteúdos de domínios, encontrando correspondência entre o domínio de trabalho e uma ontologia de sua base o agente segue para o próximo item em seu comportamento. Porém na ausência de ontologias para realizar sua operação o agente finaliza sua operação e envia uma mensagem informando que não possui recursos para auxiliar no processo.

Quando o agente encontra uma ontologia correspondente ao domínio de trabalho, realiza a operação de carregar em sua memória as informações contidas na ontologia e executa o processo de busca das informações requisitadas para aquela ontologia. Os resultados são posteriormente repassados ao agente solicitante

Figura 31 – Modelagem comportamento Agente *AOntology* com JADE/WADE.



Fonte autor (2013).

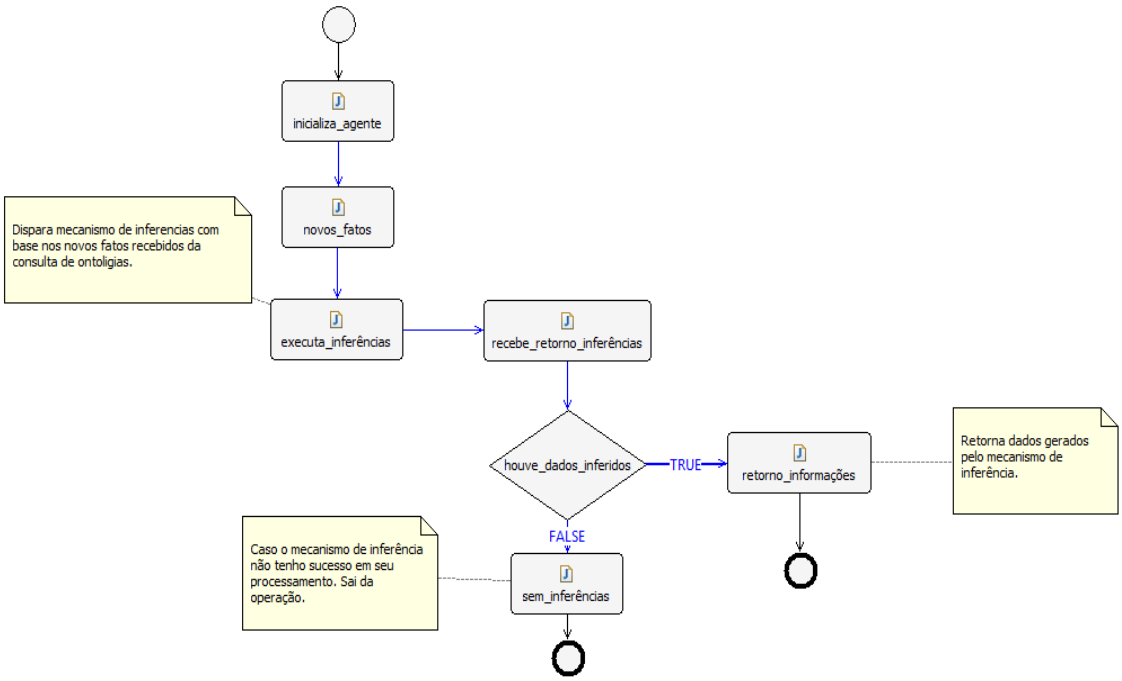
Na inexistência de informações referentes às pesquisas realizadas o agente envia a mensagem informando o insucesso da pesquisa e finaliza sua execução. Ao encerrar sua execução o agente elimina todo o conteúdo armazenado em memória para a operação de pesquisa que não obteve, ou a pesquisa que obteve sucesso. Este processo permite que enquanto o sistema *Linnaeus* esteja operando seja possível a adição de novas ontologias a base de conhecimento do sistema para posterior utilização em novas consultas.

5.5.4 Agente AEngine

O agente *AEngine* é responsável por gerenciar e executar o mecanismo de inferência (*Inference Engine*) utilizado para gerar os valores dos metadados. A figura 32 mostra a modelagem do comportamento deste agente. A operação do agente começa pela recepção de uma solicitação de valores de metadados que traz novos fatos para o mecanismo de inferência. Estes fatos são repassados ao mecanismo de inferência. Caso sejam inferidos novos valores para os metadado, então estes valores são repassados ao agente solicitante. Assim, este agente tem seu comportamento inicializado pelo recebimento de uma mensagem padrão FIPA que dispara o processo de inicialização de variáveis e alocação de memória para o recebimento dos fatos a serem transferidos ao mecanismo de inferência. No momento em que o agente esta realizando o processo de inicialização, também extraí da mensagem recebida os novos fatos e os armazena no espaço de memória alocado para este fim.

A próxima operação executada pelo agente é a transferência ou adição de novos fatos a memória de trabalho do mecanismo de inferência. Ao terminar o processo de transferência o agente dispara o mecanismo de inferência para o processamento dos novos fatos adicionados a memória de trabalho do mecanismo de inferência, para um novo processamento da base de regras por sobre estes novos fatos.

Figura 32 – Modelagem comportamento Agente *AEngine* com JADE/WADE.



Fonte autor (2013).

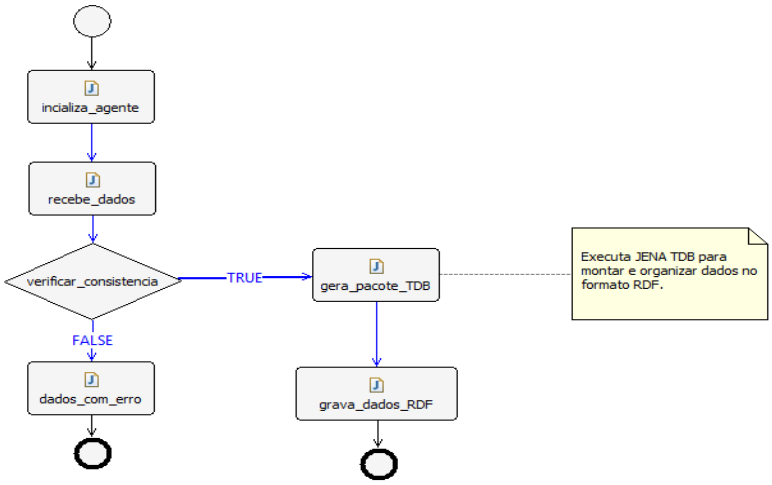
Quando o mecanismo de inferência chega ao termino de sua execução o agente *AEngine* analisa as informações inferidas pelo mecanismo, verificando se foram geradas informações que podem ser usadas como valores de metadados. Caso o mecanismo não tenha inferido nenhum novo valor, neste caso o agente envia uma mensagem para que o próximo agente solicite informações ao desenvolvedor de OA. No caso de haver valores consistentes, o agente insere estes valores nos metadados OBAA correspondentes. Ao inserir os dados inferidos nos metadados o agente verifica se existem mais campos que devem ser processados e, caso existam mais campos, o agente envia a mensagem solicitando mais informações. No momento em que todos os campos dos metadados foram preenchidos o agente informa ao próximo agente que o processo de catalogação automática chegou ao seu término.

5.5.5 Agente ADataRDF

O próximo agente pertencente ao sistema *Linnaeus* é o agente responsável pela organização e gerenciamento das informações que devem ser armazenadas na base de dados RDF do sistema. Este agente é identificado por *ADataRDF* devido ao seu acesso direto a base RDF. O comportamento do agente *ADataRDF* dentro do sistema é mostrado na figura 33, este agente possui a tarefa de manter as informações inferidas e alimentadas pelo desenvolvedor de OA integras e organizadas para seu correto armazenamento e posterior recuperação destas informações.

O agente *ADataRDF* tem seu comportamento disparado pelo recebimento da mensagem proveniente do agente *Manager* informando nesta mensagem a localização na memória das informações inferidas e o tamanho do *buffer* que contém as informações. Ao receber a mensagem o agente inicializa todas suas variáveis e parâmetros internos utilizados no processo de organização e gravação, o próximo estágio executado pelo agente é a recuperação das informações do *buffer* temporário para a sua memória de trabalho para o processo de consistência dos dados.

Figura 33 – Modelagem comportamento Agente *ADataRDF* com JADE/WADE.



Fonte autor (2013).

No processo de validação da consistência dos dados, caso de um número elevado de inconsistências nos dados o agente informa ao próximo agente no processo o agente *Manager* que não há a possibilidade da gravação das informações devido ao número elevado inconsistências nos dados e finaliza a sua operação. Já no caso dos dados consistentes e íntegros o agente passa para o próximo item de seu comportamento que se refere a criação do arquivo no formato de triplas RDF e após executa a gravação das triplas na base de dados RDF, logo após informa ao agente *Manager* que o processo de armazenamento das informações ocorreu com sucesso e finaliza sua operação.

O sistema multiagentes *Linnaeus* possui em sua estrutura de agentes a previsão para um novo agente, em que este será o responsável pela comunicação externa com outros agentes ou sistemas multiagentes. O sistema possui suporte a um ato comunicativo do tipo *propose*, que é encaminhado para o agente responsável pela operação de comunicação com os demais subsistemas da infraestrutura MILOS(GLUZ, VICARI e PASSERINO, 2012), e agentes externos que necessitem algum tipo de informação proveniente do *Linnaeus*.

5.6 Comunicação do Sistema

Esta seção apresenta as interações que ocorrem entre os agentes do *Linnaeus* e os demais componentes do sistema, em função do comportamento dos agentes definidos na seção anterior (5.5). Essas interações também foram especificadas em UML 2.1 (UML, 2013), através de diagramas de sequência (BAUER e ODELL, 2005).

Todas as mensagens provenientes do desenvolvedor de OA estão direcionadas ao agente *AManager*, o qual toma uma decisão baseado no tipo da mensagem recebida, fazendo uso de suas características de comportamento para o gerenciamento das trocas de mensagens entre os demais agentes do sistema. Assim este agente analisa as respostas geradas pelos outros agentes e quando houver necessidade de comunicação com o desenvolvedor de OA o agente realiza a comunicação com este através de *popups* com características e comportamento de interação de *wizards*, para facilitar o fornecimento de informações do usuário ao sistema.

A tabela 3 mostra todas as mensagens em uma operação de catalogação, trocadas entre os agentes *AGateway* com o desenvolvedor de OA, e as mensagens trocadas pelos o agentes *AManager*, *AOntology*, *AEngine* e o agente *ADataRDF*, onde Origem é o componente do sistema que está enviando a mensagem, Mensagem é como é chamada a mensagem, Parâmetros são as características e conteúdos contidos nas mensagens e Destino é o componente do sistema que esta recebendo a mensagem. Pode-se verificar que nenhuma mensagem enviada pelo agente *AManager* vai diretamente ao desenvolvedor de OA e nem do desenvolvedor para qualquer outro agente do sistema com exceção do agente *gateway* (como mostra a figura 24 sobre a disposição de cada componente do sistema *Linnaeus* na sessão 5.5).

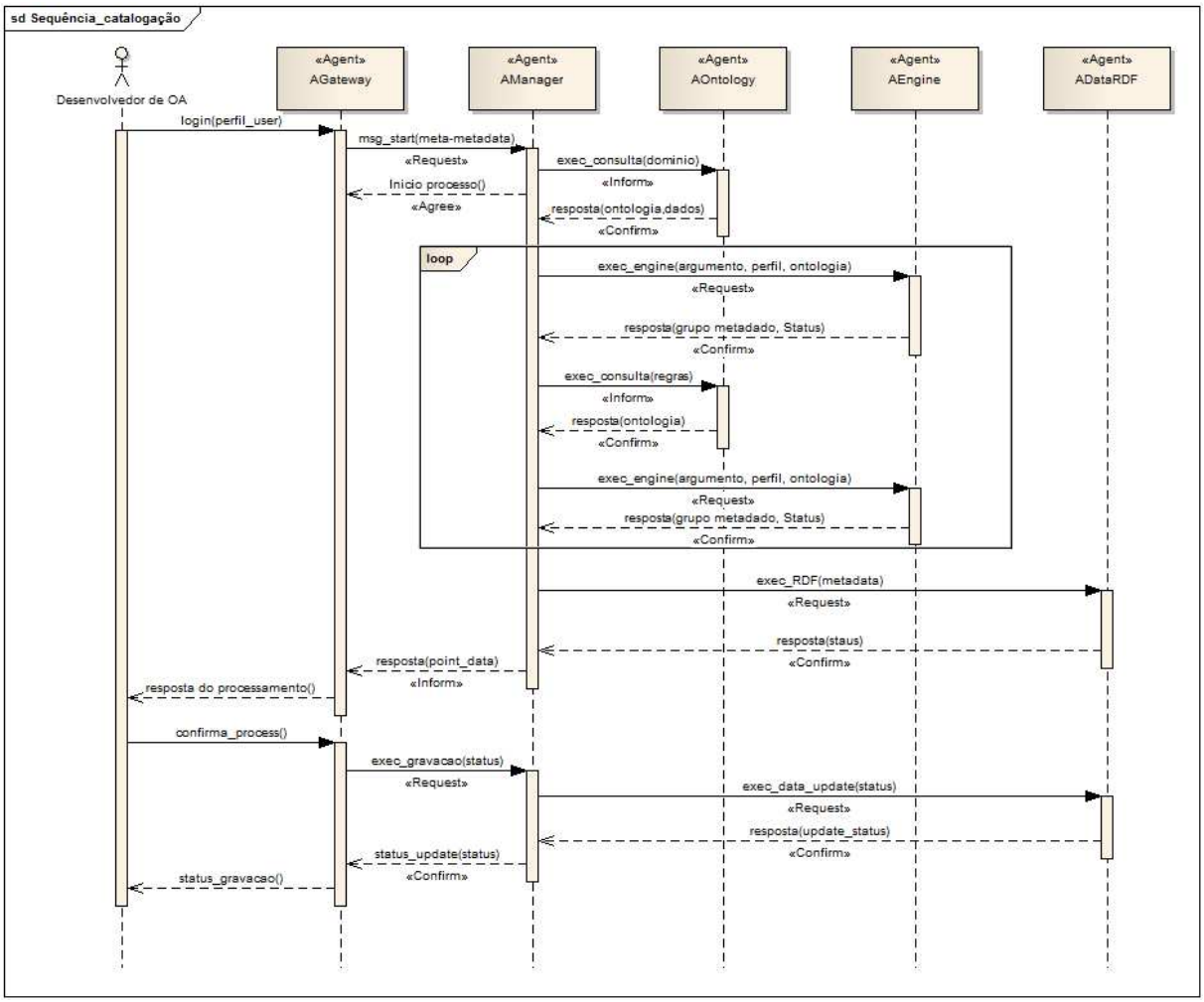
Tabela 3 - Troca de mensagens em operação de catalogação

Origem	Mensagem	Parâmetros	Destino
Desenvolvedor de OA	login	Perfil	AGateway
AGateway	msg_acesso	Argumento, Perfil	AManager
AManager	exec_consulta	Argumento, Domínio, Regras	AOntology
AManager	inicio_processo	Void	AGateway
AOntology	resposta	Ontologia, Dados	AManager
AManager	exec_engine	Argumento, Perfil, Ontologia	AEngine
AEngine	resposta	Grupo metadado, Status	AManager
AManager	exec_consulta	Argumento, Domínio, Regras	AOntology
AOntology	resposta	Ontologia, Dados	AManager
AManager	exec_engine	Argumento, Perfil, Ontologia	AEngine
AEngine	resposta	Grupo metadado, Status	AManager
AManager	exec_RDF	Metadados	ADataRDF
ADataRDF	resposta	Status	AManager
AManager	mostra_metadados	Metadados	AGateway
AGateway	resposta_process	Metadados	Desenvolvedor de OA
Desenvolvedor de OA	confrima_process	Void	AGateway
AGateway	exec_gravacao	Status	AManager
AManager	exec_data_update	Status	ADataRDF
ADataRDF	resposta	Update_Status	AManager
AManager	status_update	Status	AGateway
AGateway	status_gravacao	Void	Desenvolvedor de OA

Fonte autor (2013).

A seguir, são apresentados os diagramas de sequência, mostrando as trocas de mensagens entre os agentes e também entre o desenvolvedor de OA e o sistema. A figura 34 mostra as mensagens trocadas em um processo de catalogação automática de metadados.

Figura 34 – Diagrama de sequência para catalogação automática



Fonte autor (2013).

Pode-se ver na figura 34 que a interface do usuário informa os dados do perfil do desenvolvedor ao agente *gateway*, que por sua vez repassa as informações ao sistema de agentes com a orientação de que as informações de perfil serão utilizadas como met-metadados para o agente *manager*. O agente *manager* envia uma requisição de informações ao agente que controla a base de ontologias educacionais solicitando informações iniciais do domínio do objeto que esta sendo catalogado. Ao receber a resposta do agente *AOntology* informado o conteúdo da ontologia que deve ser utilizado no processamento o agente *manager* imediatamente envia uma requisição de processamento ao agente responsável pelo mecanismo de inferência, nesta requisição são enviados as regras para execução inicial do mecanismo juntamente com os dados recebidos da ontologia para a aplicação das regras pelo mecanismo de inferência.

O agente que executa o gerenciamento do mecanismo de inferência envia a resposta do processamento da requisição anterior contendo os elementos dos metadados preenchidos juntamente com a requisição de informações ao agente *manager*. Neste momento ocorre um processo cíclico de comunicação (*loop*) entre os agentes, *AManager*, *AOntology* e *AEngine* em função da busca de informações para na ontologia selecionada no inicio do processo, para

alimentar a base de novos fatos do mecanismo de inferência. Neste laço o agente *manager*, requisita novas informações ao agente responsável pela ontologia, e tão logo receba o retorno de sua requisição repassa as novas informações juntamente com as regras e informações para a nova execução do mecanismo de inferência. Este processo se repete até que o sistema tenha percorrido todos os itens dos metadados OBAA.

No momento em que o processo de preenchimento dos metadados é encerrado pelo mecanismo de inferência e todos os itens dos metadados OBAA foram percorridos o agente *manager* envia uma solicitação contendo os metadados para o agente responsável pelo gerenciamento da base de dados RDF. Assim que o agente *ADataRDF* recebe a primeira requisição de execução, este prepara os metadados para serem gravados em sua base e envia o retorno contendo o situação dos dados RDF gerados. O agente *manager* repassa esta situação ao agente *gateway* que por sua vez envia os dados gerados para apresentação no *display* juntamente com a solicitação de gravação ou alteração por parte do desenvolvedor.

Assim que o desenvolvedor de OA responde a solicitação do sistema, para a gravação dos metadados ao agente *gateway.* O agente *gateway* envia a solicitação de gravação ao agente *manager* que por sua vez envia a solicitação de gravação ao agente responsável pela base de dados RDF do sistema. O agente *ADataRDF* executa a gravação dos metadados na base de dados RDF e envia uma mensagem contendo a situação da gravação dos dados ao agente *manager*. O agente *manager* ajusta a mensagem contendo a situação da gravação para o entendimento do usuário e a envia ao agente *gateway* que irá transferir as informações à interface de usuário para a apresentação ao desenvolvedor de OA.

A apresentação do *status* da gravação dos metadados na base de dados RDF ao usuário encerra a sessão e a execução de todos os agentes do sistema. Deixando o sistema pronto para uma nova execução e geração de metadados.

Na execução do sistema, no momento em que o agente gerenciador do mecanismo de inferência e o agente gerenciado da base de ontologias não conseguem realizar a inferência dos metadados e extração de novas informações da base de ontologias, o agente *manager* possui em seu comportamento uma regra para a solicitação de informações do usuário através de janelas com ações do tipo *wizards*. O agente *manager* possui em sua memória duas formas pré-definidas para a solicitação de informações, estas referem-se a informações do tipo texto que o usuário informa um texto em sua resposta, e informações do tipo falso/verdadeiro em que o usuário escolhe uma opção para sua resposta.

Na tabela 4 é mostrada a sequência de mensagens trocadas em um procedimento de solicitação de informações pelo sistema ao desenvolvedor de OA. Os principais agentes no processo de interação com o usuário são os agentes *AManager* e o *AGateway.* Em que o agente *manager* recebe a solicitação do tipo de dado (*booleano, string*) que deveria ser inferido pelo sistema, com base neste parâmetro o agente monta a mensagem que será apresentada ao usuário e a repassa ao agente *gateway*. O agente *gateway* envia à requisição de uma nova janela do tipo *popup* a interface de usuário e com determinada característica. Sendo a característica uma janela do tipo *editbox* para respostas alfanuméricas, ou uma janela do tipo *radiobox* para respostas do tipo *true/false.*

Tabela 4 - Troca de mensagens para interações do tipo *wizard*

Origem	Mensagem	Parâmetros	Destino
AManager	pede_informacao	tipo_informacao	AGateway

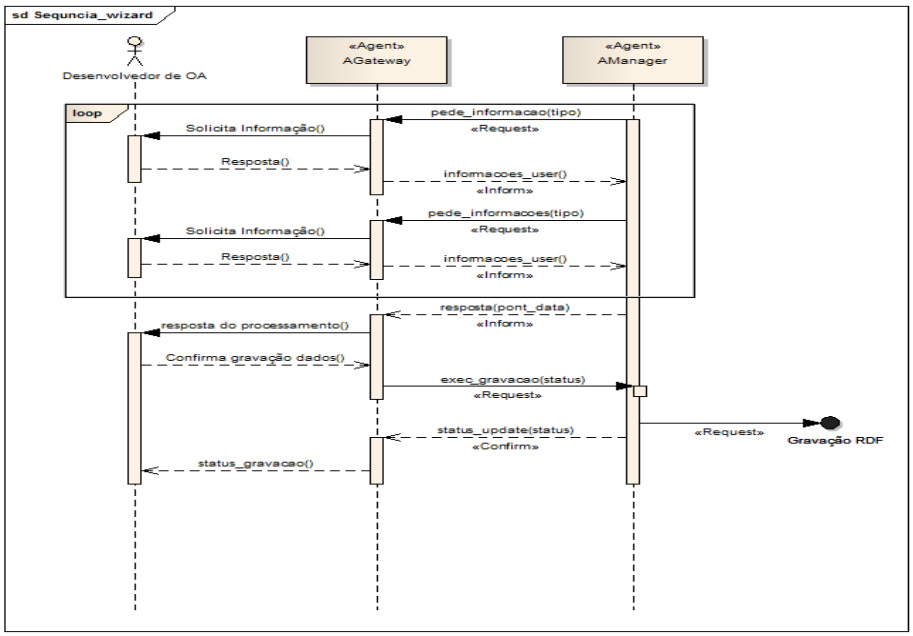
AGateway	solicita_informacao	void	Desenvolvedor de OA
Desenvolvedor de OA	resposta	void	AGateway
AGateway	informacoes_user	dados	AManager

Fonte autor (2013).

Na figura 35 é mostrado o diagrama de sequência para a operação em que o sistema solicita informações para o usuário através de janelas *popups* com o comportamento de *wizards* para o rápido entendimento do operador do sistema. O agente *manager* envia uma mensagem solicitando que o operador informe determinado do através de uma janela definida pelo parâmetro tipo informado no método de comunicação ao agente *gateway*, que ao receber esta solicitação encaminha à pergunta a interface de usuário que mostra na tela o questionamento ao usuário do sistema. O usuário responde ao questionamento feito pelo sistema, e a interface de usuário transmite as informações ao agente *gateway* que as encaminha ao agente *manager* para a continuidade do processo de preenchimento dos metadados do sistema.

O processo de interação entre os agentes do sistema e o desenvolvedor de OA pode ser executado algumas vezes, dependendo do mecanismo de inferência. No processo inferência o maior número de metadados possível deve ser preenchido pelo sistema, porém quando o mecanismo de inferência não consegue resolver uma questão, mesmo com as informações obtidas na base de ontologias o sistema deve buscar a informações do usuário. Gerando um laço de processo entre sistema e operador até que sejam percorridos todos os itens dos metadados OBAA.

Figura 35 – Diagrama de sequência para processo de solicitação de informações do usuário

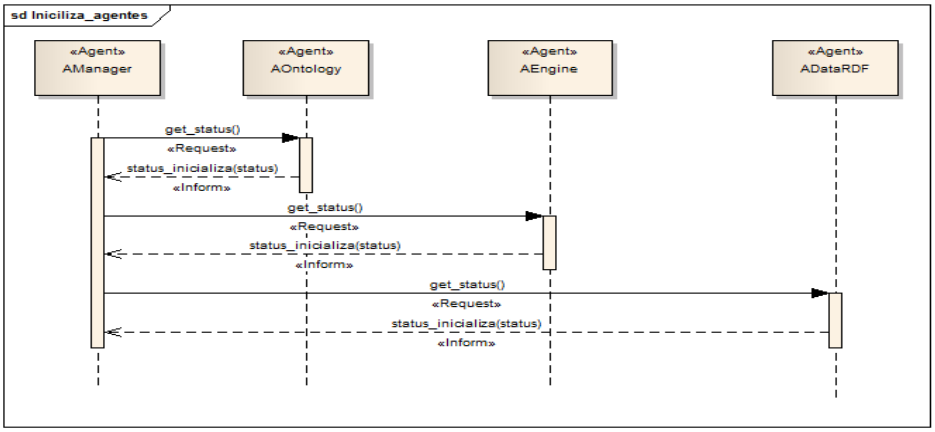


Fonte autor (2013).

Quando o mecanismo de inferência consegue chegar ao termino da execução e todos os itens dos metadados são preenchidos, o agente *AEngine* que gerencia este processo informa ao agente *manager* o termino do processo e este dá continuidade ao processo de gravação dos

dados gerados e a apresentação do metadados ao desenvolvedor de AO, como mostrado na figura 35.

Figura 36 – Diagrama de sequência para processo de avaliação do estado e inicialização dos agentes



Fonte autor (2013).

Para garantir a operação do sistema e de seus agentes, o agente *manger* envia uma requisição de informações aos agentes do sistema *Linnaues*, como mostrado na figura 36. Nesta mensagem os agentes informam seu *status* em caso de já estarem ativos ou em operação. Porém se algum dos agentes estiver em modo de *standby* a requisição enviada faz com que os agentes sejam inicializados e estejam em operação para uma operação de catalogação que possa ocorrer a qualquer momento. Proporcionando assim um rápido atendimento as requisições da operação de catalogação, fazendo com que o desenvolvedor de OA não perceba que o sistema esta inicializando.

5.7 Regras de Inferência

As ontologias de domínios educacionais desempenham um papel importante para gerar os conteúdos (valores) dos metadados dentro do sistema *Linnaeus*. O conhecimento representado na hierarquia de conceitos e nos axiomas de uma ontologia pode ser recuperado para inferir as informações que serão armazenadas como valores dos metadados. Para tanto, é necessário um mecanismo de inferência capaz de utilizar os conhecimentos relacionados aos metadados, conteúdos educacionais, informações técnicas e do usuário na produção dos valores dos metadados. De acordo com a W3C (2013), uma inferência pode ser descrita pela descoberta de novos relacionamentos entre conteúdos existentes. Nesse caso, inferência implica em um processo automático que pode gerar, através da aplicação de regras de inferência sobre os dados existentes, novas relações e fatos sobre esses dados. As novas relações são formalmente acrescentadas ao conjunto de dados inferidos até este momento.

A especificação das regras para o mecanismo de inferência tem como base os onze grupos principais dos metadados OBAA (vide sessão 5.2). Além das regras vinculadas aos grupos, também foi definido um conjunto de regras gerais para aplicação em informações provenientes do usuário.

Durante o projeto do protótipo do sistema *Linnaeus* foram analisados os mais importantes mecanismos de inferência disponíveis em JAVA: *JESS* (JESS, 2013), *JBoss*

Drools (DROOLS, 2013) e *JRuleEngine* (JRULEENGINE, 2013). O mecanismo de inferência selecionado para o sistema *Linnaeus* é o *JRuleEngine* (JRULEENGINE, 2013), devido a sua licença aberta, disponibilidade de código fonte e a praticidade no desenvolvimento das regras. Estas regras são escritas em um formato *xml* bastante flexível, podendo ocorrer a alteração, adição e exclusão de regras em tempo de execução do sistema.

A organização e elaboração das regras segue a ordem de: regras de uso geral e regras organizadas de acordo com grupos de metadados OBAA. Na Tabela 5 são apresentados exemplos de regras de uso geral em formato de linguagem natural em conjunto com as regrar equivalentes no formato *xml* utilizado pelo mecanismo de inferência *JRuleEngine*:

Tabela 5 - Regras descritas em linguagem coloquial a direita e regras JRuleEngine a esquerda

Se ocorreu download de arquivo ***então*** identifica o tipo do arquivo ***e*** armazena informações sobre arquivo*;* ***Se*** ocorreu download de arquivo do tipo vídeo flv ***então*** armazena características técnicas para este tipo de arquivo; ***Se*** ocorreu download de arquivo ***e se*** arquivo é do tipo documento pdf ***então*** armazena características técnicas do arquivo;	<?xml version="1.0" encoding="UTF-8"?> <rule-execution-set> <name>Regra de execucao 1</name> <description>Regra de preenchimento</description> <synonymn name="arquivo" class="JRuleEngine.Download_inform" /> <!-- Regras geradas para geracao semi-automatica de metadados do tipo OBAA --> <rule name="Regra1" description="Regra para obter informacoes básicas" > <if leftTerm="arquivo.getStatus" op="=" rightTerm="true" /> <then method="arquivo.procura_informacoes" /> </rule> <rule name="Regra2" description="Regra para informacoes video" > <if leftTerm="arquivo.getFormato" op="=" rightTerm="flv" /> <then method="arquivo.video_inform" /> </rule> <rule name="Regra3" description="Regra para informacoes audio tipo pdf" > <if leftTerm="arquivo.getStatus" op="=" rightTerm="true" /> <if leftTerm="arquivo.getFormato" op="=" rightTerm="pdf" /> <then method="arquivo.setAudioFileLoc" arg1="arquivo.getEndereco"/> <then method="arquivo.documento_inform" /> </rule>

Fonte autor (2013).

A seguir serão mostrados exemplos, em linguagem natural, das regras utilizadas no mecanismo de inferência para a utilização especifica de cada um dos grupos de metadados descritos pelo OBAA.

As regras para a utilização no mecanismo de inferência para o preenchimento das informações dos metadados do tipo G*eneral* são descritas abaixo;

- ***Se*** idioma preenchido ***então*** preenche campo *General.Language*;
- ***Se*** está no processo de preenchimento dados general, se usuário informou título para objeto ***então*** gera dados *General.Identifier*;
- ***Se*** informada descrição para objeto ***e se*** idioma preenchido ***então*** gera *General.KeyWord*;

As regras para a utilização no mecanismo de inferência para o preenchimento das informações dos metadados do tipo *LifeCycle* somente terão aplicação quando o desenvolvedor tiver interesse em registrar os direitos autorais do OA. Abaixo são listadas algumas regras para este grupo;

- ***Se*** campos do grupo *Rights* preenchidos ***então*** gera dados LifeCylce;
- ***Se*** campos do grupo *Rights* preenchidos ***e se*** nome do usuário informado ***então*** preenche *LifeCycle.Contribute*;

O próximo grupo com algumas de suas regras apresentadas abaixo é o grupo *Meta-Metadata*. Estes dados em sua maioria serão preenchidos pelo próprio desenvolvedor de OA no momento em que este realizar o *login* no sistema;

- ***Se*** nome do usuário informado ***e se*** dados *General* preenchidos ***então*** preenche campo *Meta-Metadata.Contribute* ***e*** preenche campo *Meta-Metadata.Role*;
- ***Se*** nome do usuário informado ***e se*** idioma informado ***então*** preenche campo *Meta-Metadata.Language*;

Os metadados do grupo *technical*, quando houver o *download* de um arquivo devem ser preenchido no momento posterior ao termino do processo de *download.* Algumas regras para este grupo de metadados é apresentada abaixo;

- ***Se*** arquivo armazenado é de vídeo ***então*** preenche campo *Technical.Duration*;
- ***Se*** arquivo armazenado ***então*** preenche campo *Technical.SpecificLocation* ***e*** preenche campo Technical.SpecificSize;

Quando o desenvolvedor de OA informar o domínio educacional do objeto, então os os metadados do tipo *educational* podem ser preenchidos com base em informações adicionais extraídas das ontologias de conteúdo. Abaixo algumas regras para o preenchimento destes metadados;

- ***Se*** domínio do OA informado ***e se*** aplicação do OA especificada ***então*** preenche *Educational.IntendedEndUserRule*;
- ***Se*** domínio do OA informado ***e se*** o idioma informado ***então*** preenche campo *Educational.Language*;
- ***Se*** domínio do OA informado ***e se*** OA possui observações ***então*** preenche campo *Educational.Description*;

Se o desenvolvedor de OA tiver o interesse em registrar a autoria e direitos do objeto catalogado, os metadados *rights* devem ser preenchidos.

- ***Se*** usuário registrou direito autoral ***então*** preenche campos *Rights*;

A relação de um documento com outro arquivo ou documento, para a criação de um objeto de aprendizagem, devem ter suas informações descritas nos metadados do tipo *relation.* Abaixo algumas regras para o preenchimento deste metadados;

- ***Se*** OA possui mais de um documento ***então*** preenche campo *Relation.Kind*;
- ***Se*** OA possui mais de um documento ***e se*** OA relacionado existe na base de dados ***então*** preenche campo *Relation.Resouce*;

Os metadados do tipo *annotation*, tem um exemplo de regra para preenchimento de seus metadados listado abaixo;

- *Se* nome do usuário informado ***e se*** data de catalogação informada ***então*** preenche campos *Annotation*;

O metadados do tipo *classification* devem ser preenchidos de acordo com o domínio do objeto e suas características. Abaixo são listadas algumas regras para o preenchimento deste grupo de metadados;

- ***Se*** OA com domínio informado ***então*** preenche campo *Classification.Purpouse*;
- ***Se*** OA com domínio informado ***e se*** campo *Classification.Purpouse* preenchido ***então*** gera dados *Classification.TaxonPath*;

Os metadados referentes ao grupo *acessibility*, tem algumas regras listadas abaixo;

- ***Se*** arquivo armazenado é vídeo ***então*** preenche campo *Acessibility.hasVisual* ***e*** preenche campo *Acessibility.ResourceDescription*;
- ***Se*** arquivo armazenado é áudio ***então*** preenche campo *Acessibility.hasAuditory* ***e*** preenche campo *Acessibility.ResourceDescription*;

As regras referentes ao grupo *SegmentInformationTable*, tem um exemplo mostrado abaixo;

- ***Se*** arquivo armazenado é vídeo ***e se*** arquivo esta divido em partes ***então*** preenche campos *SegmentInformationTable*.

5.8 Base de Dados RDF

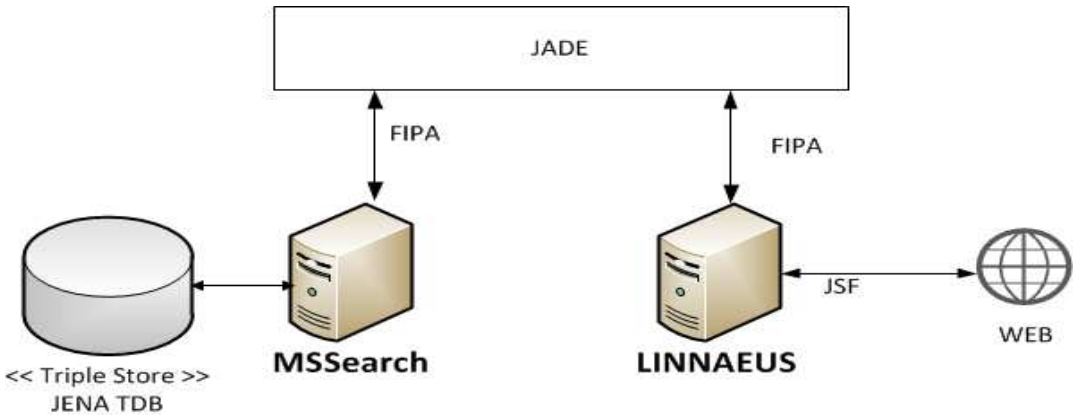
No sistema *Linnaeus*, a base de dados semântica que será utilizada para armazenar os metadados gerados para os OAs e realizar consultas para utilização nas inferências é o repositório semântico *MSSearch* (Da Silva, 2013). Esta base de dados também é utilizada por outros sistemas para a recuperação de objetos de aprendizagem, ou para consultas e avaliação dos objetos nela contidos.

A base de dados semântica dispõe todos os serviços de armazenamento e recuperação dos metadados OBAA, utilizados pelos agentes do sistema. A característica semântica desse repositório se deve ao suporte nativo ao formato OWL, tanto para fins de consulta, através da linguagem de consulta SPARQL, quanto para fins de armazenamento em triplas RDF.

O acesso ao repositório semântico *MSSearch* se dará pelo protocolo Jade/FIPA. Utilizando a linguagem SPARQL para realizar o acesso a base semântica através dos agentes responsáveis pelo acesso a base RDF/TDB no sistema *MSSearch*. O comando contendo a requisição no formato SPARQL enviada a base semântica, esta encapsulado em uma mensagem Jade/FIPA com o ato comunicativo do tipo inform. Este ato comunicativo é proveniente do próprio repositório semântico *MSSearch*, sendo apenas utilizado pelos agentes do *Linnaeus* para acesso a base de dados semântica.

Na figura 37 é mostrado o fluxograma da interação realizada pelo *Linnaeus* com o repositório semântico para a gravação e recuperação de informações na base de dados. Com a visão de que os sistemas estão distribuídos em uma rede local, o *Linnaeus* através de sua plataforma Jade conecta-se ao *main container* do repositório semântico para o processo de interação entre os sistemas.

Figura 37 – Conexão com repositório semântico



Fonte autor (2013).

Mensagem FIPA com o comando SPARQL encapsulado em seu conteúdo é recebida pelo repositório semântico, e este conteúdo é processado na base semântica. Um exemplo SPARQL para busca de informações, este exemplo de busca permite descobrir quais objetos estão contidos no repositório pode ser implementada pela seguinte consulta SPARQL (Figura 38):

Figura 38 – Consulta SPARQL para recuperação de OAs

```
SELECT ?lobj
      WHERE {?lobj a LearningObject}
```

Fonte: Da Silva (2013).

O resultado da consulta do exemplo, deve ter como retorno a identificação de todos os objetos pertencentes a classe *LearningObject* presentes no repositório. Na tabela 6 é mostrado o resultado da operação de consulta executada, pelo comando de exemplo mostrado anteriormente.

Tabela 6 - Resultado comando exemplo de consulta SPARQL

lobj
lo001
lo002
lo003
lo004

Fonte autor (2013).

6 EXPERIMENTOS E VALIDAÇÕES

Este capítulo apresenta os experimentos realizados para a avaliação do protótipo da sistema *Linnaeus*. O objetivo dos experimentos foi validar aspectos da usabilidade deste sistema e da qualidade dos metadados inferidos pelo mesmo. Para a usabilidade considerou-se como métrica objetiva de avaliação o número de interações do usuário com o sistema para a geração do conjunto de metadados de um dado OA. Nesta métrica, assume-se que um menor número de interações corresponde a um melhor índice de usabilidade. O número de interações do sistema Linnaeus é então comparado com o número de interações necessárias para criar os mesmos metadados através de uma interface de edição de metadados sem nenhum suporte inteligente.

Em termos de qualidade dos metadados gerados foi efetuada uma comparação dos metadados gerados para um dado conjunto de OA, com os objetos equivalentes obtidos do BIOE (Banco Internacional de Objetos Educacionais)[2] em um domínio de ensino específico. Para a realização destes experimentos foram utilizados dez (10) objetos de aprendizagem provenientes do BIOE. A fonte de objetos BIOE foi escolhida devido a sua já consolidada utilização em âmbito nacional por várias instituições de ensino, fazendo assim com que seu conteúdo possa ser utilizado com métrica de avaliação de qualidade para os objetos gerados pelo *Linnaeus*.

6.1 Avaliação do processo de interação

Na avaliação da usabilidade do *Linnaeus* foram analisados a quantidade de interações realizadas pelo sistema com o usuário para a criação dos metadados do OA que esta em processo de catalogação. O número de interações pode indicar o quão eficiente é o sistema para a realização da catalogação do metadados do OA. Esta eficiência será medida pela quantidade de interações do sistema com o número de metadados gerados pelo sistema.

O Linnaeus possui dois tipos de interação com usuário para o recebimento de informações, que são a apresentação de janelas de diálogo *popups* do tipo *editbox* para a entrada de texto e *popups* do tipo *radiobox* para o recebimento de informações do tipo VERDADEIRO/FALSO. Apesar dessa interação ocorrer na forma de janelas de diálogos, estas formas de interação são essencialmente equivalentes aos campos de edição de textos ou de múltiplas opções, disponíveis para edição dos metadados através de uma interface de formulários similar a interface disponível no DSpace (ver Capítulo 4). A diferença com o *Linnaeus*, que será medida neste experimento, é que para criar o mesmo número de metadados no DSpace, ou no modo *expert* do *Linnaeus*, será necessário preencher (e portanto interagir) com um número de entradas no formulário no mínimo igual ao número de metadados a serem criados, exigindo no mínimo uma interação para o preenchimento de cada um dos metadados do OA.

A tabela 7 mostra os resultados do experimento que gerou novamente os metadados dos 10 objetos previamente escolhidos do BIOE, mas desta vez com o suporte dos *wizards* de

2 O BIOE (http://objetoseducacionais2.mec.gov.br/) é um repositório de OA com o aval do governo brasileiro que é mantido através do Ministério da Educação (MEC) e do Ministério da Ciência e Tecnologia (MCT) e que contém OA de acesso público, em diversos formatos e para todos os níveis de ensino.

apoio a catalogação disponíveis no sistema Linnaeus. O domínio educacional destes objetos é a matemática para o ensino médio, desta forma o sistema utilizará a ontologia MilosEduMat-v10.owl[3] juntamente com a ontologia OBAA22.owl[4] para a inferência dos metadados.

Tabela 7 - Quantidade de Interações x Metadados inferidos

Objeto	Número de Interações	Quantidade Metados Gerados	% Interações em Modo Especialista
a_altura_da_arvore	11	47	23%
A_Criacao_dos_Logaritmos_Parte_I	13	51	25%
A_Criacao_dos_Logaritmos_Parte_II	13	51	25%
Apostas_no_relógio	11	43	25%
Building_curves_hyperbola	15	48	31%
funcao_afim	14	45	31%
Geometria	15	53	28%
metadados_3x+1_part1	14	54	25%
metadados_3x+1_part2	14	54	25%
metadados_Patron_de_cylindre	17	52	32%
TOTAL	**137**	**498**	**27%**

Fonte autor (2013).

De acordo com os dados apresentados, o sistema teve um bom desempenho na sua tarefa de catalogação semiautomática de metadados. Por exemplo, o OA "metadados_3x+1_part1" com um total de quatorze (14) interações foram inferidos pelo sistema cinquenta e quatro (54) metadados. Desta forma se um desenvolvedor de OA realizasse o processo manual de catalogação deste mesmo objeto, ele realizaria cinquenta e quatro (54) interações para preencher o mesmo número de metadados gerados pelo *Linnaeus*. A última coluna da tabela 7 compara, através de percentuais, o número de interações com suporte de *wizards*, com o número de interações equivalentes em modo *expert*. Assim, com o suporte dos *wizards* do *Linnaeus* o desenvolvedor somente precisaria efetuar 25% das interações necessárias no modo expert.

Estes resultados permitem concluir que o sistema *Linnaeus*, quando operando com o suporte de *wizards*, está solicitando um número baixo de interações para o preenchimento de uma quantidade significativa de metadados. Como mostrado na tabela 7 o número de interações realizadas pelo sistema para a solicitação de informações ao desenvolvedor de OA foi relativamente baixa em relação ao número de metadados gerados. A média de 27% de interações obtidas pela interface com *wizards* do *Linnaeus* é bastante significativa, implicando

[3] http://obaa.unisinos.br/MilosEduMat-v10.owl

[4] http://obaa.unisinos.br/obaa22.owl

em um número baixo de interações quando comparado com o número de interações equivalentes no modo expert de interface (ou então usando uma ferramenta como o DSpace).

6.2 Avaliação metadados gerados pelo sistema

A avaliação da qualidade dos metadados gerados pelo sistema *Linnaeus* foi realizada de forma comparativa com os metadados dos objetos obtidos do BIOE. Os objetos de aprendizagem obtidos do BIOE tem a organização de seu metadados, com a utilização do metadado Dublin Core (DCMI, 2012). Como o *Linnaeus* utiliza o padrão de metadados OBAA, nessa análise comparativa foi utilizado o perfil de compatibilização OBAA - DCMI não-qualificado definido no "Relatório Técnico RT-OBAA-01" (VICARI; GLUZ, 2009), adaptado para os novos metadados DCMI utilizados no BIOE.

Para a aplicação deste teste no sistema foram recuperados dez (10) OA do BIOE do domínio educacional matemática para ensino médio, cujos identificadores no BIOE são:

- "a_altura_da_arvore"
- "A_Criacao_dos_Logaritmos_Parte_I"
- "A_Criacao_dos_Logaritmos_Parte_II"
- "Apostas_no_relógio"
- "Building_curves_hyperbola"
- "funcao_afim"
- "Geometria"
- "metadados_3x+1_part1"
- "metadados_3x+1_part2"
- "metadados_Patron_de_cylindre".

Esta análise comparativa tem sua finalidade em comparar o conteúdo gerado pelo sistema com o catalogo original do objeto, indicando assim uma medida da qualidade das regras de inferência e do conhecimento utilizado para a realização do processo de catalogação, para os metadados do domínio educacional de matemática do ensino médio.

Na tabela 8 são apresentados os metadados do OA "metadados_3x+1_part1", tal como catalogados no BIOE. Os metadados deste objeto, assim como dos demais OA, foram utilizados como referência para a análise da qualidade dos metadados gerados pelo *Linnaeus* .

Tabela 8 - Metadados objeto de aprendizagem BIOE: "mat6_32-3x+1-BL1.mp3"

Campo Dublin Core	Valor	Idioma
dc.audience.mediator	Secretaria de Educação Básica (SEB/MEC)	pt_BR
dc.audience.educationlevel	Ensino Médio	pt_BR
dc.description.tableofcontents	Ensino Médio:Matemática	pt_BR

dc.type	Áudio	pt_BR
dc.language	Pt	pt_BR
dc.location.country	Br	pt_BR
dc.subject.category	Educação Básica::Ensino Médio::Matemática::Números e operações	pt_BR
dc.description.abstract	O programa apresenta a conjectura do problema 3x+1 e discute algumas curiosidades em torno dela para mostrar que, mesmo parecendo verdade, os matemáticos só consideram verdadeiro aquilo que é provado lógica e matematicamente	pt_BR
dc.description2	1. Apresentar a conjectura de Collatz 2. Mostrar um aspecto formal da matemática em um problema simples de entender	pt_BR
dc.rightsholder	MEC	pt_BR
dc.rights.license	Termo de cessão dado pelo autor ou seu representante diretamente ao Ministério da Educação - MEC que permite o uso do recurso para distribuição, tradução, edição, excetuando-se o uso comercial	pt_BR
dc.relation.requires	MAT 6.32-3X+1-BL1.mp3	*
dc.date.submitted	2012-01-16T11:20:30Z	*
dc.date.accessioned	2012-01-16T11:20:30Z	
dc.date.available	2012-01-16T11:20:30Z	
dc.identifier.uri	http://objetoseducacionais2.mec.gov.br/handle/mec/20188	
dc.contributor.author	Projeto Condigital MEC - MCT	
dc.contributor.author	Universidade Estadual de Campinas - Unicamp - Matemática	
dc.date.issued	2012-01-16T11:20:30Z	
dc.subject.keyword	Números Naturais	pt_BR
dc.subject.keyword	Conjuntos	pt_BR
dc.contributor.author	Oliveira, Samuel Rocha de	
dc.contributor.author	Fujihira, Vanessa	
dc.contributor.author	Oliveira, Samuel Rocha de	
dc.contributor.author	Sarti, Luis Ricardo	
dc.date.created	2012-01-16	
dc.subject.keyword	Lógica	pt_BR
dc.subject.keyword	Conjectura	pt_BR
dc.title	3x + 1 - Parte I	pt_BR
dc.relation.isreferencedby		
collection.collection.collection	/handle/mec/154||Ensino Médio: Matemática: Áudios	

Fonte BIOE (2013).

A tabela 9 mostra o mapeamento dos metadados DCMI estendido utilizados pelo BIOE nos metadados OBAA funcionalmente equivalentes. Dentro do possível, todas as equivalências entre metadados DCMI estendido e metadados OBAA foram identificadas e mapeadas na tabela 9.

Tabela 9 - Metadados OBAA compatíveis com metadados DCMI usados no BIOE

Dublin Core (BIOE)	OBAA
dc.audience.mediator	obaa.MetaMetadata.Contribute.Role
dc.audience.educationlevel	obaa.educational.Context
dc.description.tableofcontents	obaa.general.description
dc.type	obaa.technical.Requirement.OrComposite.Type
dc.language	obaa.general.Language

dc.location.country	
dc.subject.category	obaa.educational.Context
dc.description.abstract	obaa.educational.Description
dc.description2	
dc.rightsholder	obaa.rights
dc.rights.license	
dc.relation.requires	obaa.MetaMetadata.Identifier.Catalog
dc.date.submitted	obaa.MetaMetadata.Contribute.Date
dc.date.accessioned	
dc.date.available	
dc.identifier.uri	obaa.technical.Location
dc.contributor.author	obaa.MetaMetadata.Contribute
dc.contributor.author	
dc.date.issued	obaa.MetaMetadata.Contribute.Date
dc.subject.keyword	obaa.general.keyword
dc.subject.keyword	
dc.contributor.author	obaa.MetaMetadata.Contribute
dc.contributor.author	
dc.contributor.author	
dc.contributor.author	
dc.date.created	obaa.lifecycle.Contribute.Date
dc.subject.keyword	obaa.general.keyword
dc.subject.keyword	
dc.title	obaa.general.Title
dc.relation.isreferencedby	obaa.relation
collection.collection.collection	obaa.classification

Fonte Autor (2013).

Foi identificado um total de vinte (20) metadados DCMI estendido, utilizados pelo BIOE, que são compatíveis com o padrão de metadados OBAA. Um dos metadados utilizado pelo BIOE não obteve correspondência no OBAA, este metadado se refere ao campo “dc.location.country”.

Na tabela 10 são apresentados os metadados inferidos pelo sistema *Linnaeus* para o objeto BIOE “metadados_3x+1_part1” apresentado na tabela 8. Como pode-se ver nesta tabela, os metadados gerados pelo *Linnaeus* cobrem grande parte dos itens existentes no BIOE.

Tabela 10 - Metadados OBAA gerados pelo sistema Linnaues

OBAA	Metadado
General	
obaa.general.Identifier	http://objetoseducacionais2.mec.gov.br/handle/mec/20188
obaa.general.Title	Objeto para ensino de sistemas lineares

obaa.general.Language	ptBR
obaa.general.description	Objeto para Matematica_Ensino_Medio , com Conteudo_Matematica_Ensino_Medio em Sistemas_Lineares na área de Equacoes_Lineares
obaa.general.keyword	Matematica_Ensino_Medio , Conteudo_Matematica_Ensino_Medio, Sistemas_Lineares, Equacoes_Lineares
LifeCycle	
obaa.lifecycle.Version	1.0
obaa.lifecycle.Status	final
obaa.lifecycle.Contribute	Ederson Silveira
obaa.lifecycle.Contribute.Role	Desenvolvedor de OA
obaa.lifecycle.Contribute.Entity	Unisinos
obaa.lifecycle.Contribute.Date	2013-10-04T17:09:04Z
MetaMetadata	
obaa.MetaMetadata.Identifier	http://objetoseducacionais2.mec.gov.br/handle/mec/20188
obaa.MetaMetadata.Identifier.Catalog	Linnaeus_obaa_mat6_32-3x+1-BL1.mp3
obaa.MetaMetadata.Identifier.Entry	
obaa.MetaMetadata.Contribute	Ederson Silveira
obaa.MetaMetadata.Contribute.Role	Desenvolvedor de OA
obaa.MetaMetadata.Contribute.Entity	Unisinos
obaa.MetaMetadata.Contribute.Date	2013-10-04T17:09:04Z
obaa.MetaMetadata.MetadataSchema	
obaa.MetaMetadata.Language	ptBR
Technical	
obaa.technical.Format	mp3
obaa.technical.Size	11,550512
obaa.technical.Location	http://objetoseducacionais2.mec.gov.br/handle/mec/20188
obaa.technical.Requirement	
obaa.technical.Requirement.OrComposite	
obaa.technical.Requirement.OrComposite.Type	reprodutor de audio
obaa.technical.Requirement.OrComposite.Name	VLC audio
obaa.technical.Requirement.OrComposite.MinimumVersion	pc, placa de som, player mp3
obaa.technical.Requirement.OrComposite.MaximumVersion	
obaa.technical.InstallationRemarks	
obaa.technical.OtherPlatformRequirements	
obaa.technical.Duration	06:09:00

obaa.technical.SupportedPlatforms	Aplicavel em todos os sistemas operacionais, e players de audio como mp3player.
obaa.technical.PlatformSpecificFeatures	
obaa.technical.PlatformSpecificFeatures.PlatformType	
obaa.technical.PlatformSpecificFeatures.SpecifcFormat	audio mp3
obaa.technical.PlatformSpecificFeatures.SpecificSize	11,550512
Educational	
obaa.educational.InteractivityType	baixo
obaa.educational.LearningResourceType	audio narrativo
obaa.educational.InteractivityLevel	baixo
obaa.educational.SemanticDensity	
obaa.educational.IntendedEndUserRole	Aluno
obaa.educational.Context	Objeto para Matematica_Ensino_Medio , com Conteudo_Matematica_Ensino_Medio em Sistemas_Lineares na área de Equacoes_Lineares
obaa.educational.TypicalAgeRange	16-18
obaa.educational.Difficulty	facil
obaa.educational.TypicalLearningTime	PT6M09S
obaa.educational.Description	Objeto para Matematica_Ensino_Medio , com Conteudo_Matematica_Ensino_Medio em Sistemas_Lineares na área de Equacoes_Lineares
obaa.educational.Language	ptBR
obaa.educational.LearningContentType	Conteudo_referente_a_conceito
obaa.educational.Interaction	
obaa.educational.Interaction.Perception	audio narrativo
obaa.educational.Interaction.Synchronism	VERDADEIRO
obaa.educational.Interaction.CoPresence	FALSO
obaa.educational.Interaction.Reciprocity	01-01
obaa.educational.DidacticStrategy	Para estratégia_pedagogica o objeto deve ser aplicado como exercicio em laboratório
Rights	
obaa.rights.Cost	licença GPL
obaa.rights.CopyrightAndOtherRestrictions	
obaa.rights.Description	Utilização em ensino de mátemática
Relation	
obaa.relation.Kind	
obaa.relation.Resource	

obaa.relation.Resource.Identifier	
obaa.relation.Resource.Identifier.Catalog	
obaa.relation.Resource.Identifier.Entry	
obaa.relation.Resource.Description	
Annotation	
obaa.annotation.Entity	Ederson Silveira
obaa.annotation.Date	2013-10-04T17:09:04Z
obaa.annotation.Description	Material audio para ensino matematica
Classification	
obaa.classification.Purpose	
obaa.classification.TaxonPath	http://obaa.unisinos.br/MilosEduMat-v10.owl
obaa.classification.TaxonPath.Source	MilosEduMat-v10.owl
obaa.classification.TaxonPath.Taxon	
obaa.classification.TaxonPath.Taxon.Id	
obaa.classification.TaxonPath.Taxon.Entry	
obaa.classification.Description	
obaa.classification.Keyword	
Acessibility	
obaa.acessibility.ResourceDescription	
obaa.acessibility.ResourceDescription.Primary	
obaa.acessibility.ResourceDescription.Primary.HasVisual	
obaa.acessibility.ResourceDescription.Primary.HasAuditory	VERDADEIRO
obaa.acessibility.ResourceDescription.Equivalent.Content.AlternativesToVisual.AudioDescription	Material para resolucao de matematica ensino medio
obaa.acessibility.ResourceDescription.Equivalent.Content.AlternativesToVisual.AudioDescription.Language	ptBR

Fonte Autor (2013).

A tabela 11 apresenta a comparação dos metadados obtidos do BIOE com relação aos metadados inferidos pelo sistema *Linnaeus* para todos os 10 objetos de teste. Estes dados totalizam a existência de informações geradas pelo *Linnaeus* na forma de metadados OBAA equivalentes aos metadados DCMI empregados no BIOE, tal como definido pela tabela 9. Para a geração dos percentuais foi considerado o valor de total de vinte e um (21) metadados distintos utilizados no BIOE para a catalogação dos 10 OA de teste.

Tabela 11 - Resultado metadados gerados relacionados com metadados BIOE

Objeto	Total de Metadados Gerados	OBAAxBIOE (%)
a_altura_da_arvore	18	86%
A_Criacao_dos_Logaritmos_Parte_I	17	81%
A_Criacao_dos_Logaritmos_Parte_II	17	81%

Apostas_no_relógio	19	90%
Building_curves_hyperbola	16	76%
funcao_afim	19	90%
Geometria	18	86%
metadados_3x+1_part1	20	95%
metadados_3x+1_part2	20	95%
metadados_Patron_de_cylindre	16	76%
TOTAL	**10**	**86%**

Fonte Autor (2013).

Os resultados da tabela 11 mostram que o *Linnaeus* é capaz de gerar de forma semi-automática 76% a 95% dos metadados utilizados pelo BIOE. Combinado com os resultados anteriores (tabela 7), que estes metadados são gerados em média com menos de 30% das interações da interface disponível no DSpace, isso indica um bom ganho no processo de catalogação destes objetos.

Em relação à qualidade dos metadados individuais, deve-se ressaltar que, apesar de não ter sido feita uma comparação formal da semântica dos metadados gerados pelo *Linnaeus* com a semântica dos metadados equivalentes no BIOE, porque tal comparação formal foge muito ao escopo do presente trabalho, foi feita uma comparação informal da qualidade destes metadados. Essa qualidade depende de três fatores: (a) a ontologia do domínio de ensino, (b) as regras de inferência de metadados e (c) as respostas fornecidas pelo usuário. Comparando-se a tabela 8 com a tabela 10 pode-se verificar que, no caso dos metadados técnicos e educacionais foram inferidas, graças a análise de formato de conteúdo, várias das propriedades relacionadas à compatibilidade com sistemas operacionais, tipo de interação possível, modo de uso em sala de aula, além de aspectos de acessibilidade.

Em termos da qualidade da descrição, que é obtida em grande parte pela interação de inferências sobre a ontologia de domínio de ensino, com respostas do usuário, pode-se observar que as diferenças do texto descritivo gerado pelo *Linnaeus* “Objeto para Matematica_Ensino_Medio, com Conteudo_Matematica_Ensino_Medio em Sistemas_Lineares na área de Equacoes_Lineares” com o texto do BIOE “O programa apresenta a conjectura do problema 3x+1 e discute algumas curiosidades em torno dela para mostrar que, mesmo parecendo verdade, os matemáticos só consideram verdadeiro aquilo que é provado lógica e matematicamente” podem ser classificadas em termos de generalidade: o texto gerado pelo *Linnaeus* apresenta o contexto geral de um problema de equação linear (no caso a equação “3x+1”). O texto do BIOE não faz esta contextualização, mas mostra detalhes específicos que o professor (ou designer de conteúdo) quis associar ao seu objeto.

É importante ressaltar que mesmo na interface com suporte de *wizards* é possível acrescentar textos com esse tipo de informações adicionais. Porém, como no caso dos testes com os 10 objetos do BIOE, o usuário do *Linnaeus* não era professor nem designer de conteúdos para o domínio de ensino em questão (Matemática para Ensino Médio), não foi acrescentada nenhuma informação particularizada ao objeto.

Dessa forma, os experimentos realizados com os objetos de aprendizagem obtidos do BIOE mostram evidência que o sistema *Linnaeus* está conseguindo atingir seu objetivo de oferecer um apoio inteligente para processo de catalogação de OA, permitindo que esse processo possa ser feito de forma mais fácil e rápida, sem haver uma perda significativa na quantidade e qualidade dos metadados gerados.

7 CONSIDERAÇÕES FINAIS

Este capítulo tem por objetivo apresentar as considerações finais sobre o projeto desenvolvido. Para tanto, os objetivos traçados inicialmente são recapitulados, as etapas desenvolvidas até a obtenção dos resultados finais são brevemente apresentadas. Além disso, a seção trabalhos futuros apresenta sugestões de tópicos a serem desenvolvidos visando a melhoria deste trabalho, bem como a sua integração a outros temas de pesquisa.

7.1 Considerações

A dissertação apresentou as principais características da ferramenta *Linnaeus*, cujo protótipo tem como objetivo verificar a viabilidade da questão de pesquisa dissertação. Como objetivos específicos a serem alcançados durante o desenvolvimento do sistema estavam:

- Analisar as soluções atuais de catalogação de OA a fim de especificar um processo de catalogação baseado em ontologias educacionais e no uso de *wizards* de catalogação capazes de interpretar os conhecimentos dessas ontologias para ajudar no processo de catalogação
- Projetar um sistema para catalogação formado por agentes especialistas na elaboração, criação e alteração de metadados e por agentes *wizards* capazes de aplicar os conhecimentos adquiridos de ontologias de domínios educacionais no processo de catalogação inteligente;
- Desenvolver um protótipo desse sistema que possa operar em um ambiente *web*;
- Planejar e conduzir experimentos e testes de forma que os métodos utilizados na catalogação possam ser avaliados e validados;

O objetivo de analisar ferramentas de catalogação, ou com suporte a catalogação de OA. Com a finalidade de especificar um novo processo de catalogação do metadados OBAA(GLUZ, VICARI e PASSERINO, 2012), foi atingida com a união de todas as tecnologias apresentadas nesta dissertação para a criação do *Linnaeus.*

O sistema *Linnaues* foi desenvolvido como um sistema multiagente utilizando a plataforma JADE/FIPA(BELLIFEMINE F.; G. CAIRE, 2002), com uma interface *Web*, capaz de apoiar as atividades de criação e edição dos metadados OBAA dos OAs para o domínio educacional. A organização das camadas da aplicação foi feita da seguinte forma: na interface com o usuário foi utilizada a tecnologia JAVA JSF, o acesso às ontologias foi feito através de agentes JADE com o auxílio do *framework* JENA (JENA, 2012), para sistema de raciocínio dos agentes foi utilizado a ferramenta *JRuleEngine* (JRULEENGINE, 2013), e os metadados gerados são enviado para a base de dados *MSSearch* (Da Silva, 2013).

A criação e desenvolvimento do protótipo do sistema para testes, também foi atingida com o sistema desenvolvido no decorrer da presente dissertação o sistema *Linnaues*. A interface *web* foi projetada para que possa ocorrer a interação dos agentes com o usuário do sistema através de janelas do tipo *popups web*.

O Capítulo 6 mostra os experimentos conduzidos com o protótipo do sistema *Linnaeus*. Os resultados apresentados são evidências que o sistema *Linnaeus* conseguiu atingir seu objetivo de oferecer um apoio inteligente para processo de catalogação de OA,

permitindo que esse processo possa ser feito de forma mais fácil e rápida, sem haver uma perda significativa na quantidade e qualidade dos metadados gerados.

Das pesquisas empreendidas durante a dissertação foram produzidos dois artigos científicos publicados em anais de eventos. Este dois artigos apresentam análises sobre a evolução das pesquisas sobre as tecnologias do sistema e as características deste. O artigo "Sistema LINNAEUS: apoio inteligente para a catalogação e edição de metadados de objetos de aprendizagem" de SILVEIRA e GLUZ foi publicado nos Anais do SBIE 2012. O artigo foi reimpresso no Anexo I.

O segundo artigo "Uma ferramenta para fornecer apoio a catalogação de metadados de objetos de aprendizagem - LINNAEUS" de SILVEIRA, GALÃO e GLUZ, também será publicado nos anais do SBIE 2013. Este artigo, que foi reimpresso no Anexo II, apresenta uma versão inicial da arquitetura do Sistema *Linnaeus*, que é a ferramenta computacional de auxilio semiautomático ao processo de criação de metadados OBAA empregada nessa dissertação.

7.2 Trabalhos Futuros

O sistema de catalogação *Linnaeus* oferece suporte web para a catalogação de objetos de aprendizagem que possa se juntar a uma solução de maior âmbito que compreenda um ambiente de autoria, recuperação e consulta de objetos de aprendizagem (GLUZ e VICARI, 2010; GLUZ, VICARI e PASSERINO, 2012). A integração do sistema de catalogação com um sistema de autoria de conteúdos deve ser o próximo passo seguido para o aprimoramento do sistema *Linnaeus* e de seus agentes. Em um próximo passo a integração do *Linnaeus* com o sistema de autoria de conteúdo para ter a sua totalidade de serviços testados e validados pelo usuário final.

No decorrer do desenvolvimento deste projeto percebeu-se que, os dois tipos de informações solicitadas ao desenvolvedor de OA, podem ser expandidas para novos tipos de informações e requisições. Desta forma sugere-se que para implementações futuras a criação de janelas possibilitando múltiplas escolhas por parte do usuário sejam projetadas. Também o desenvolvimento de classe para representar e receber informações mais complexas presentes nos metadados OBAA, como metadados estruturados.

A realização de experimentos com ontologias de outros domínios de ensino pode oferecer uma melhor visão da capacidade de aquisição de conhecimento do sistema, e sua inferências neste cenário. O mecanismo de inferência utilizado no sistema também pode ser aprimorado com a adição de novas regras para tratar da análise de vários formatos de conteúdos.

A arquitetura do sistema *Linnaeus* prevê a utilização de um agente que será responsável pela comunicação com os demais sistemas que necessitarem informações ou até mesmo tiverem a intenção de utilizar o *Linnaeus* como parte de suas operações para a realização de autoria de objetos. Entretanto o agente que deve realizar a comunicação com sistemas externos e a comunicação com os agentes existentes, deve ser desenvolvido e integrado ao *Linnaeus*.

Por fim a interface *web* do sistema pode ser melhorada e aprimorada para atender ao diferentes tipos de perfis de usuários do sistema. Em que cada perfil tenha a interface

modelada para a melhor interação entre sistema e usuário, e o recebimento de informações do usuário possa também ter suas características baseadas no perfil do usuário.

REFERÊNCIAS

ADOBE - **ACTIONSCRIPT TECHNOLOGY CENTER**. Disponível em: < http://www.adobe.com/devnet/actionscript.html/>. Acesso em: mai. 2013.

BARBONE,V.G.; RIFON,L.A.,**From SCORM to Common Cartridge: A step forward**, 2009.Em Journal Computers & Education, 2010.88-102.

BAUER, B.; MULLER, J.; ODELL, J**. Agent UML: A formalism for specifying multiagent software systems**. Int. J. Softw. Eng. Knowl, 2001.207--230.

BAUER, B.; ODELL, J. **UML 2.0 and Agents: How to Build Agent-based Systems with the new UML Standard**. 2002. Em Journal of Engineering Applications of AI, Volume 18, pp 141-157.

BALL,S., TENNEY, J. **Xerte – A User-Friendly Tool for Creating Accessible Learning Objects,** 2008. Em Proceedings of the International Conference on Computers Helping People with Special Needs. Lecture Notes in Computer Science 5105, pp. 291 a 294.

BERGENTI, F.; POGGI, A. **Exploiting UML in the Design of Multi-agent Systems**. 2000. Proceedings of the First International Workshop on Engineering Societies in the Agent World: Revised Papers, p.106-113.

BELLIFEMINE, F. CAIRE G., POGGI A., RIMASSA G. **JADE – A White Paper.** 2003. Disponivel em: <http://jade.tilab.com/papers/2003/WhitePaperJADEEXP.pdf>. Acesso em: dez. 2012.

BELLIFEMINE, F., CAIRE, G. **JADE Programmer's Guide.** Jun 2007. Disponível em: <http://www.jade.tilab.com>. Acesso em: set. 2012 .

BELLIFEMINE, F. POGGI, A. **JADE – A FIPA-Compliant Agent Framework**. PAAM'99. London: 97-108.

BERGENTI, F., CAIRE, G. **Interactive Workflows with WADE.** 2012. Disponível em: <http://jade.tilab.com/wade/papers/ACEC-2012.pdf>. Acesso em: fev. 2013.

BIOE. **Banco Internacional de Objetos Educacionais (BIOE) Normas para a definição dos metadados**. Disponivel em: <http://objetoseducacionais2.mec.gov.br/retrievefile/normas>. Acesso em: abr 2013.

BIZER, C., HEATH, T., BERNERS-LEE, T. **LINKED DATA - THE STORY SO FAR**. International Journal on Semantic Web and Information Systems (IJSWIS). 5(3):1 – 22, 2009.

BEZ, M. R., FLORES, C. D., ZANATTA, E., FRAZÃO, S., ROEHE, A. e VICARI, R. M. (2010) "**Banco de imagens médicas para desenvolvimento de material pedagógico**". In: XXI Simpósio Brasileiro em Informática na Educação, João Pessoa.

BRASIL. Ministério da Educação. **Secretaria de Educação a Distância. Objetos de aprendizagem: uma proposta de recurso pedagógico**. Organização: PRATA, C., NASCIMENTO, A. C. – Brasília: MEC, SEED, 2007. 154 p.

BREITMAN, K.K. ; SAMPAIO DO PRADO LEITE, J.C. **Ontology as a requirements engineering product.** Requirements Engineering Conference, 2003. Proceedings. 11th IEEE International.

CAIRE, G. **WADE User Guide.** – Jun 2011. Disponível em: < http://jade.tilab.com/wade/doc/WADE-User-Guide.pdf >. Acesso em fev. 2013.

CAIRE, G., GOTTA, D., BANZI, M. **WADE: A software platform to develop mission critical applications exploiting agents and workflows**. AAMAS'08.Portugal: 29-36.

CASTRO, J.; ALENCAR, F.; SILVA, C. **Engenharia de Software Orientada a Agentes**. Publicado em Livro das Jornadas de Atualização em Informática – JAI, capitulo 5 2006.

COLLIS, J.; NDUMU, D.; NWANA, H.; LEE, L. **The ZEUS Agent Building Tool-kit**. Publicado em BT Technology Journal, Springer Netherlands, vol. 16, pp 60 – 68, 1998.

DA SILVA, Luiz Rodrigo Jardim**. MSSearch: Busca semântica de objetos de aprendizagem obaa com suporte a alinhamento automático de ontologias**, Brasil. 2013. 92 f. Dissertação (Mestrado em Computação Aplicada) – Programa de Pós-Graduação em Computação Aplicada. Universidade do Vale do Rio dos Sinos, São Leopoldo, RS, 2013.

DCMI. **Dublin Core Metadata Element Set**, Version 1.1. 2008. Disponível em: < http://dublincore.org/documents/dces/ >. Acesso em: set. 2012.

DIAS, C. L., KEMCZINSKI, A. L., SÁ, S. V., FERLIN, J., HOUNSELL, M. S. **Padrões abertos: aplicabilidade em Objetos de Aprendizagem (OAs)**. XX Simp. Bras. de Inf. na educação. SBIE 2009.

DROOLS, Drools - The Business Logic integration Platform, 2013. Disponivel em: < http://www.jboss.org/drools/ >. Acessado em: jul. 2013.

DUTRA, R. L., TAROUCO, L. **Objetos de Aprendizagem: Uma comparação entre SCORM e IMS Learning Design**. Disponível em: < http://www.lume.ufrgs.br/handle/10183/13028 >. Acesso em: out. 2012.

EISENBERG, A.; MELTON, J.; KULKARNI, K.; MICHELS, J.; ZEMKE, F. **SQL:2003 Has Been Published**. SIGMOD Record 2004;33(1):119–126.

FIPA. **FIPA Specifications - Repository**. Disponível em: <http://www.fipa.org/specs/fipa00023/SC00023K.pdf>. Acesso em: set. 2012.

FOWLER, M. **UML Distilled**. 2000. Massachusetts. Addison Wesley.

FRANKLIN, S. & GRAESSER, A. **Is it an agent, or just a program?: A taxonomy for autonomous agents**. In Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, pages 21–35, Berlin, Germany. Springer Verlag, 1996.

GLUZ, J. C.; VICARI, R. M**. MILOS: Infraestrutura de Agentes para Suporte a Objetos de Aprendizagem OBAA**. Anais do XXI SBIE, 2010.

GLUZ. J. C. **INTRODUÇÃO A INFRAESTRUTURA MILOS**. Pós-Graduação em Computação Aplicada (PIPCA), Universidade do Vale do Rio dos Sinos (UNISINOS). Maio, 2010.

GLUZ, J. C.; VICARI, R. **Uma Ontologia OWL para Metadados IEEE-LOM, Dublin-Core e OBAA**. Anais do SBIE 2011, Aracaju, 2011. v. 1. p. 204-213.

GLUZ, J.C., VICARI, R e PASSERINO, L. **An Agent-based Infrastructure for the Support of Learning Objects Life-Cycle**. Procs. of ITS 2012, Chania, Creta, 2012. Lecture Notes in Computer Science. New York: Springer, 2012. v. 7315. p. 691-693.

GLUZ, J.C., VICARI, R e PASSERINO, L. **An OWL Ontology for IEEE-LOM and OBAA Metadata.** Procs. of ITS 2012, Chania, Creta, 2012. Lecture Notes in Computer Science. New York: Springer, 2012. v. 7315. p. 696-698.

GÓMEZ-PEREZ, et. al. (2004**). Ontological Engineering**, Madrid: Springer, pp. 3-45.

GRIFFIN, N. L., LEWIS, F. D. **A Rule-Based Inference Engine which is Optimal and VLSI Implementable.** Disponível em: < http://www.cs.uky.edu/~lewis/papers/inf-engine.pdf >. Acesso em: mar. 2013.

GUARINO, N. **Formal Ontology and Information Systems.** Proceeding of the FOIS' 98. Trento, Italy, 1998. IOS Press.

IEEELTSC. Std1484.12.**1 IEEE Learning Technology Standard Committee (LTSC) Standard for Learning Object Metadata (LOM)**. IEEE, 2002.

JENA. **Framework for Building Semantic Web Applications**. Disponível em: <http://jena.apache.org/>. Acesso em: set. 2012.

JENNINGS, N.; SYCARA, K.; WOOLDRIDGE, M. **A Roadmap of Agent Research and Development**. 1998. Em Journal Autonomous Agents and Multi-Agent Systems, Holanda.

JENNINGS, N. R.; WOOLDRIDGE, M**. J. Intelligent Agents: Theory and Practice. Knowledge Engineering** Review, v; 10 n. 2, 1995.

JRULEENGINE, **JRuleEngine is java rule engine**, 2013. Disponivel em: < http://jruleengine.sourceforge.net >. Acessado em: jun. 2013.

JESS. **The Rule Engine for the Java Platform**. 2008. Disponível em: < http://www.jessrules.com >. Acesso em: fev. 2013.

KRATZ, R. A., CRESPO, S. SCOPEL, M., BARBOSA, J. **Fábrica de Adequação de Objetos de Aprendizagem.** Revista Brasileira de Informática na Educação. Volume 15 – Número 3 – Setembro a Dezembro de 2007.

LIND, J. **Agent Oriented Software Engineering**. 2006. Disponível em:< http://www.agentlab.de/aose.html >. Acesso em: jul. 2012.

NOY, N. F., MCGUINNESS D. L. **Ontology Development 101: A Guide to Create Your First Ontology**. Stanford University Conference – 2004.

SACCHI, G., MARANDO, A., QUARANTOTTO, E., CAIRE, G. **WADE Tutorial.** 2011. Disponível em: < http://jade.tilab.com/wade/doc/tutorial/WADE-Tutorial.pdf >. Acesso em: mar. 2013.

SIRIN, E., PARSIA, B., GRAU, B. C., KALYANPUR, A., KATZ, Y. **PELLET: A PRATICAL OWL-DL REASONER**, Journal of Web Semantics: Science, Services and Agents on the World Wide Web. Valencia 2007. v. 5, p. 51-53.

SMITH, M., BASS,M., MCCLELLAN, G., TANSLEY, R., BARTON, M., BRANSCHOFSKY, M., STUVE, D., WALKER, J. H.,**DSpace Na Open Source Dynamic Digital Repository**, 2003, D-LIB Magazine.

UML. **Documents Associated With UML Version 2.1.2**. Disponível em: <http://www.omg.org/spec/UML/2.1.2/>. Acesso em: fev 2013.

USCHOLD, M., JASPER, R. **A framework for understanding and classifying ontology applications**.In Procs of IJCAI 99 Workshop on Ontologies and Problem-Solving Methods (KRR5), Stockholm, Sweden, 1999.

VICCARI, R.; GLUZ, J. **Infraestrutura OBAA-MILOS: Infraestrutura Multiagente para Suporte a Objetos de Aprendizagem OBAA – Rel. Técnico – Especificação Detalhada da Arquitetura da Infraestrutura MILOS**. UFRGS/CINTED, 2009.

VICCARI, R.; GLUZ, J. PENTEADO, F. **Infraestrutura OBAA-MILOS: Infraestrutura Multiagente para Suporte a Objetos de Aprendizagem OBAA – Rel. Técnico – Especificação Preliminar da Arquitetura do Sistema de Apoio Pedagógico.** UFRGS/CINTED, 2009.

VICCARI, R.; GLUZ, J.; PASSERINO, L.; et al. **The OBAA Proposal for Learning Objects Supported by Agents**. Procs. of MASEIE Workshop – AAMAS 2010, Toronto, Canada, 2010.

VICCARI, R.; GLUZ, J.; SANTOS, E.; et al. **Projeto OBAA – Rel. Técnico RTOBAA01 – Proposta de Padrão para Metadados de Objetos de Aprendizagem Multiplataforma**. UFRGS/CINTED, 2009. Disponível em: <http://www.portalobaa.org/obaac/padraoobaa/relatoriostecnicos/RTOBAA01.pdf/view>.

W3C. **RIF RDF e OWL Compatibilidade.** Disponível em: <http://www.w3.org/TR/2010/REC-rif-rdf-owl-20100622/>. Acesso em: dez. 2011.

W3C. **Vocabularies.** Disponível em: <http://www.w3.org/standards/semanticweb/ontology>. Acesso em: jan. 2013.

W3C. **World Wide Web Consortium (W3C)**. Disponível em: <http://www.w3.org/>. Acesso em: out. 2012.

WILEY, David A. **Connecting Learning Objects to Instructional Design Theory: A Definition, A Metaphor**, and A Taxonomy. 2001. In: WILEY, D. **The Instructional Use of Learning Objects: Online Version.** Disponível em http://reusability.org/read/chapters/ wiley.doc. Acessado em: ago. 2012.

WOOLDRIDGE, M.; JENNINGS, N. **Agent Theories, Architectures, and Languages: A Survey**. 1994. Amsterdam, Holanda. Em Workshop on Agent Theories, Architectures and Languages. p. 1-32.

WOOLDRIDGE, M. **An Introduction to Multiagent Systems**. John Wiley & Sons, 2002.

ANEXO I - ARTIGOS PUBLICADOS

Anais do XXIII SBIE - XVIII CBIE Rio de Janeiro, 26 a 30 de novembro de 2012. Sistema LINNAEUS: apoio inteligente para a catalogação e edição de metadados de objetos de aprendizagem.

Sistema LINNAEUS: apoio inteligente para a catalogação e edição de metadados de objetos de aprendizagem

Ederson Luiz Silveira[1], João Carlos Gluz[1]

[1]PIPCA – Programa Interdisciplinar de Pós-Graduação em Computação Aplicada– Universidade do Vale dos Sinos (UNISINOS)
Av. Unisinos, 950 – 93.022-000 – Cristo Rei – São Leopoldo – RS – Brasil

esilveira1@gmail.com, jcgluz@unisinos.br

Abstract. *Learning objects are assuming a greater role in the educational context, providing digital resources that are increasingly present in classrooms and in distance education. This increasing role creates the necessity of tools able to help users to catalog and/or edit the metadata of these objects. To provide support to this process this paper proposes the system LINNAEUS, an intelligent tool, which supports learning object cataloging, and metadata editing activities. The paper presents the main features of LINNAEUS, and use cases of the system.*

Resumo. *Os objetos de aprendizagem vêm tendo um papel crescente no contexto educacional, oferendo recursos digitais que estão cada vez mais presentes em salas de aula e na educação a distância. Com sua crescente utilização, surge à necessidade de ferramentas que auxiliem o processo de catalogação e de edição dos metadados destes objetos. Para prover apoio a este processo, o presente trabalho propõe uma ferramenta inteligente de suporte a catalogação de objetos de aprendizagem e edição de metadados, o sistema LINNAEUS. No trabalho são apresentadas as principais características do LINNAEUS, além de casos de uso do sistema.*

1. Introdução

Com o crescente emprego da internet em todos os âmbitos do cotidiano, além da grande variedade existente de ferramentas direcionadas ao ensino, aumenta a possibilidade de tornar a mediação digital entre professor e aluno mais didática e produtiva. Um Objeto de Aprendizagem (OA) é mais um recurso didático digital para auxiliar professores e alunos. Entretanto, para que este OA possa ser armazenado, localizado e ser alterado remotamente pelo professor, este deve ter suas informações de catalogação devidamente preenchidas e registradas de acordo com o domínio do objeto, a aplicação, a sua localização na Web, a plataforma de utilização, dentre outras informações. No contexto da padronização dos OA, tal catalogação é implementada pelos metadados.

Este artigo apresenta o sistema de catalogação LINNAEUS, cujo objetivo é prover uma solução de software que dê apoio à autoria de objetos de aprendizagem descritos e suportados pela proposta para o padrão de metadados OBAA [Vicari e tal. 2010; Bez et. al

2010]. O padrão OBAA é definido com base no padrão IEEE-LOM [IEEE-LTSC, 2002] incluindo suporte para a adaptabilidade e interoperabilidade entre OAs em diversas plataformas de operação como Web, TV Digital, dispositivos móveis, a sua compatibilidade com padrões internacionais, a acessibilidade aos OAs por todas as pessoas inclusive as portadoras de necessidades especiais, e a independência juntamente com flexibilidade do padrão que não necessita de tecnologias proprietárias, e que permita que inovações tecnológicas sejam acrescidas ao padrão, sem perder a compatibilidade com o material já desenvolvido [Bez et al., 2010].

Os OA podem ser vistos como artefatos descritos em duas camadas (ou níveis):

- Camada dos Metadados: que engloba as informações de catálogo do objeto de aprendizagem, informando sua localização na base de dados, domínio do objeto, aplicação, plataforma de operação, etc... Dentre as informações com significativa importância na aplicação educacional destacam-se dados descritivos utilizados em busca, localização, recuperação e apresentação do conteúdo;
- Camada de Conteúdo: que contém o material de aprendizado que deve ser visualizado pelo usuário para atingir os objetivos de determinado tópico de ensino.

O sistema LINNAEUS opera somente na camada de metadados. O protótipo deste sistema será integrado à infraestrutura MILOS [Vicari e Gluz, 2010], fornecendo parte do serviço de autoria de OA a ser disponibilizado pela infraestrutura. Assim o LINNAEUS efetua um intercâmbio de informações com os demais subsistemas da infraestrutura.

A organização deste artigo é a seguinte: a Seção 2 e 3 apresentam o referencial teórico do artigo. A Seção 4 arquitetura do sistema e dos seus componentes, na Seção 5 é apresentada à aplicação de exemplo da ferramenta e na seção 6 são apresentas as considerações finais.

2. Catalogação de Objetos de Aprendizagem

Metadados são informações que descrevem os objetos de aprendizagem, assim sempre que houver uma alteração no conteúdo do OA é importante que isso se reflita nos seus metadados, causando uma atualização dos mesmos para manter coerentes as informações de catálogo a respeito de um objeto de aprendizagem.

Porém a complexidade dos padrões de metadados para OA existentes atualmente IEEE-LOM e DCMI força que a criação ou alteração de metadados requeira um alto nível de conhecimento técnico desses padrões [Gluz e Vicari, 2010]. Uma possível solução para esse problema é oferecer um suporte inteligente para as tarefas de criação, atualização ou adição de metadados. Para realizar esta tarefa é de grande importância que e ferramenta tenha o conhecimento do conjunto de metadados disponibilizado pelo padrão escolhido, além de conhecimentos sobre como eles devem ser configurados a partir do surgimento de novas versões adaptadas de um mesmo conteúdo.

Com o objetivo de maximizar a aderência aos padrões de metadados existentes atualmente e garantir a compatibilidade dos metadados, este trabalho irá utilizar somente metadados compatíveis com o padrão OBAA, porque este padrão engloba os padrões de metadados IEEE-LOM e DCMI [Vicari e tal. 2010, Bez et al., 2010], além de permitir o tratamento de questões de acessibilidade e interoperabilidade entre plataformas digitais.

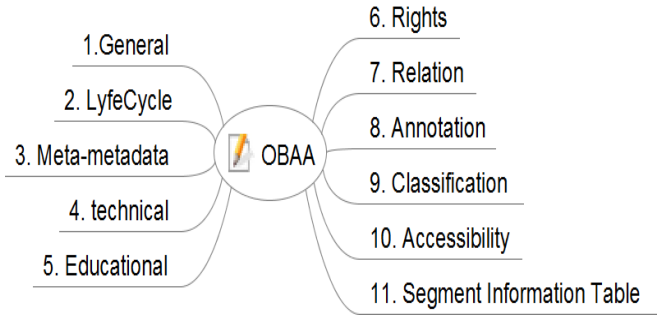


Figura 1 – Classificação dos grupos de Metadados OBAA. Fonte autor.

Dessa forma, o objetivo do presente trabalho é projetar e desenvolver um sistema multiagente capaz de manipular os metadados do padrão OBAA e realizar as alterações do conteúdo correspondente. Para tanto será definida uma arquitetura de um sistema multiagente integrado à infraestrutura MILOS [Vicari, Gluz e Passerino, 2012; Gluz e Vicari, 2010].

3. Infraestrutura MILOS

O projeto de pesquisa OBAA-MILOS tem por objetivo a busca da convergência das tecnologias de objetos de aprendizagem (OA) e sistemas multiagente, para auxiliar no desenvolvimento de ambientes de aprendizagem. Tais objetos de aprendizagem são baseados na tecnologia de agentes inteligentes para seu desenvolvimento.

O resultado concreto deste projeto será a infraestrutura MILOS que oferecerá suporte a objetos de aprendizagem compatíveis com o padrão OBAA, fornecendo suporte ao ciclo de vida inteiro de um objeto de aprendizagem e possibilitando que estes objetos sejam interoperáveis na WEB, televisão digitais e dispositivos móveis. A infraestrutura MILOS faz uso extensivo das tecnologias de ontologias e de agentes para desenvolver aplicações inteligentes de manipulação de objetos de aprendizagem. Agentes inteligentes de software, com suporte de mecanismos de inferência capazes de manipular os tipos de representações de conhecimento usados em ontologias (OWL), serão os principais componentes ativos da infraestrutura MILOS. MILOS [Vicari, Gluz e Passerino, 2012; Gluz e Vicari, 2010].

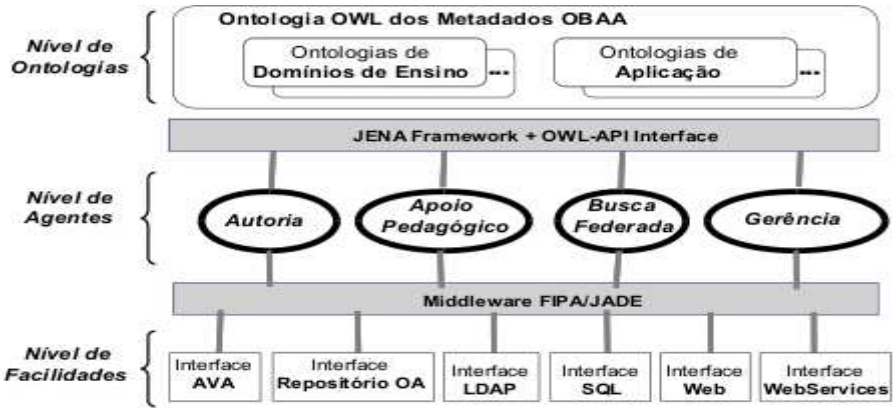


Figura 2 – Organização geral da MILOS. Fonte [Gluz e Vicari, 2010].

A arquitetura da infraestrutura MILOS é dividida em três grandes níveis de abstração conforme mostrado na Figura 2: (a) Nível das Ontologias: responsável pela especificação do conhecimento que será compartilhado entre os agentes da infraestrutura; (b) Nível de Agentes: responsável pela implementação do suporte aos requisitos de adaptabilidade, interoperabilidade e acessibilidade previstos na proposta OBAA; (c) Nível das Facilidades de

Interface: responsável pela comunicação dos agentes da MILOS com servidores Web, ambientes virtuais, repositórios de objetos de aprendizagem, bancos de dados, serviços de diretórios e demais tipos de aplicações educacionais. As principais entidades de cada nível da infraestrutura MILOS, respectivamente ontologias, agentes e facilidades, desempenham papéis organizacionais específicos para que a arquitetura da MILOS possa operar.

4. Sistema *LINNAEUS*

O nome LINNAEUS é inspirado em Carl Linneaus criador da taxonomia moderna e catalogação [Koerner, 1999]. Tendo essa inspiração em mente, espera-se que a principal característica do LINNAEUS será auxílio o projetista (designer) ou desenvolvedor de objetos de aprendizagem a catalogar de forma rápida e fácil os novos objetos. Assim o sistema LINNAEUS será encarregado de editar os metadados dos objetos de aprendizagem suportados pela plataforma MILOS.

A proposta do sistema apresentado no presente artigo é o desenvolvimento de um sistema para catalogação de OA de forma inteligente e automático de grande parte do conteúdo dos metadados suportados pela proposta OBAA-MILOS, exigindo do usuário final nenhum ou o conhecimento básico em relação a objetos de aprendizagem. Este sistema será desenvolvido utilizando à linguagem de programação JAVA com suporte a web, atendendo umas premissas da proposta OBAA-MILOS de que suas ferramentas devem ser *open source* de domínio publico. O sistema LINNAEUS possui suas operações e tarefas implementadas por meio de agente de software [Wooldridge, 2002; Weiss, 1999], usando o conceito de agentes *wizards* inteligentes. A criação do catalogo de metadados do sistema se dará de forma automática sem o conhecimento do usuário, o catalogo seguirá como uma de suas premissas o domínio do OA, sua aplicabilidade, plataforma de operação, para a criação do catalogo.

Cada grupo de agente *wizard* é responsável pela realização de uma tarefa dentro do sistema. Essas tarefas podem consistir: em analisar o tipo de objeto de aprendizagem que será criado, buscando identificar a finalidade do OA (qual o objetivo do desenvolvedor para este objeto), a metodologia de aplicação do OA e a área deste objeto dentro do domínio educacional.

Os agentes do sistema LINNAEUS são organizados em uma hierarquia. No nível mais elevado de hierarquia há um agente chamado “coordenador do sistema”. Neste agente são centralizados todos os comandos e orientações para que os grupos de agentes situados em um nível hierárquico abaixo possam realizar a execução de tarefas e ações de acordo com a necessidade específica do desenvolvedor de OA, o domínio do objeto de aprendizagem, sua utilização e aplicação e a plataforma digital em que será utilizada. Este conjunto de agentes pode ser visto como uma distribuição em formato de pirâmide, de forma que o agente gerenciador de ambiente e coordenação esta no localizado no topo e os demais agentes localizados nos níveis abaixo. O agente “coordenador do sistema” tem como seu objetivo prover informações de orientação e de operação para a camada inferior de agentes. Suas intenções dentro do sistema são de recolher um determinado número de informações recebidas do desenvolvedor de OA para que possa atingir seu objetivo. A arquitetura da figura 3 é composta de agentes que executam seus papéis de forma atômica e autônoma. Os principais componentes da arquitetura são descritos a seguir:

O agente “gerenciador de ambiente” è responsável criação da tarefa macro para os agentes *wizards*. Uma tarefa macro consiste em preparar o ambiente para o desenvolvedor, ajuste dos itens que serão solicitados para o usuário do sistema, o pré-preenchimento dos metadados quando o perfil do usuário for iniciante ou intermediário.

Na camada de agentes abaixo do agente “gerenciador do sistema”, encontram-se os agentes *wizards* responsáveis pela execução de partes de uma tarefa macro dentro do sistema. Tarefas como a criação de interface para o operador, tarefas de comunicação com os demais sistemas da MILOS através do protocolo FIPA, tarefas de elaboração e criação de informações para a catalogação de metadados de acordo com o domínio do objeto de aprendizagem, sua aplicação, plataforma de operação.

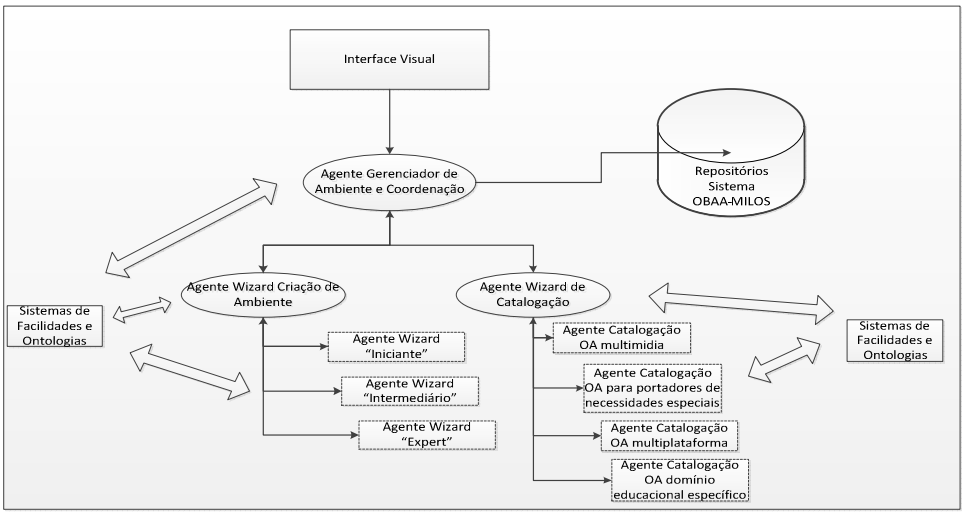


Figura 3 – Arquitetura Proposta para o Sistema. Fonte: autor

O *wizard* de coordenação da catalogação necessita das informações descritas pelo usuário e que indicam qual é o domínio de ensino do objeto de aprendizagem e qual o tipo de aplicação ou uso educacional que se pretende dar ao objeto. Elas são usadas para identificar as ontologias e agentes *wizards* de catalogação específicos para o objeto. Logo após obter essas informações, o agente passa a coordenar os trabalhos dos agentes especialistas em catalogar aspectos específicos do OA. A divisão de trabalhos destes agentes especialistas de catalogação é baseada na estrutura dos metadados OBAA. São previstos agentes especialistas para aplicações educacionais que envolvem o uso de objetos de aprendizagem multimídia, objetos para portadores de necessidades especiais, objetos de aprendizagem que operam em mais de um sistema operacional, objetos de aprendizagem de domínios educacionais específicos (matemática, logica, algoritmos e estruturas de dados, etc..).

O objetivo do *wizard* de criação de ambiente é gerar e fornecer uma interface para o desenvolvedor utilizando as características do perfil e do conhecimento técnico do usuário. Este agente possui três opções para criação de interface para o desenvolvedor, estas seguem o conhecimento do desenvolvedor em objetos de aprendizagem. O primeiro nível é o nível leigo, caracterizando-se por usuários que possuem pouco ou nenhum conhecimento técnico sobre objetos de aprendizagem. Para este nível o *wizard* irá fornecer uma interface com questões dividida em três etapas e finalizará a catalogação do objeto de aprendizagem sem que o usuário tenha a conhecimentos técnicos sobre objetos de aprendizagem.

O nível intermediário assume que o usuário possui um conhecimento técnico básico sobre metadados dos OAs, desta forma o *wizard* irá fornecer ao desenvolvedor uma interface também com perguntas relacionadas ao domínio de ensino do objeto, a aplicação e demais características, com a opção do desenvolvedor de OA poder visualizar os metadados se for de seu interesse, esta operação de catalogação é divida em três ou quatro etapas dependendo da necessidade do desenvolvedor. Já o último nível “expert”, o *wizard* apresenta todos os

campos pertinentes ao domínio de ensino, aplicação do objeto, plataforma de operação, para que o desenvolvedor possa alimentar estas informações de acordo com seu conhecimento.

Figura 4 – Tela inicial *LINNAEUS*. Fonte autor.

A figura 4 apresenta a tela do protótipo do sistema de catalogação LINNAEUS, na em sua tela principal os agentes *wizard* desenvolveram a interação com o desenvolvedor de objetos de aprendizagem para receber as informações necessárias para o preenchimento dos metadados do objeto utilizado pelo desenvolvedor.

5. Cenário de Uso do LINNAEUS

Para a apresentação do cenário de uso do sistema LINNAEUS, a seguir será mostrado um exemplo da sua utilização com um objeto de aprendizagem para o ensino de matemática.

O proposito do usuário apresentou-se ao sistema como tendo pouco ou nenhum conhecimento técnico sobre OAs, tendo a intenção de criar um objeto de aprendizagem com domínio no ensino de matemática. Este OA é do tipo multimídia para auxiliar na tutoria de equações e sistemas trigonométricos. Após a autoria do conteúdo do objeto o sistema LINNAEUS é executado e como o conhecimento técnico do usuário é pouco ou nenhum em relação à OAs o sistema apresentará as informações previamente preenchidas por ele para que o usuário possa altera-las ou seguir para o próximo passo se for o desejo do usuário.

Na figura 5 é mostrada a primeira tela apresentada ao usuário para a catalogação do objeto de aprendizagem, com o preenchimento prévio dos campos de acordo com as informações passadas pelo usuário na autoria do conteúdo. Caso o desenvolvedor desejar pode alterar as informações contidas nos campos, ou seguir para a próxima etapa do processo de catalogação.

Figura 5 – Tela que solicita os dados básicos de identificação do objeto de aprendizagem (*wizard* para o nível iniciante). Fonte autor.

Com a intenção de agilizar o processo de catalogação de objetos de aprendizagem, o sistema LINNAEUS nos modos iniciante e intermediário realiza o preenchimento automático de grande parte dos metadados principais utilizados para o determinado OA solicitando apenas as informações de maior importância.

As próximas figuras apresentam a catalogação de um OA para o domínio de ensino na área da matemática. Com o perfil do desenvolvedor selecionado para iniciante, como pode ser visualizado na figura 5, por se tratar de um perfil com pouco conhecimento em objetos de aprendizagem o *wizard* minimiza a solicitação de informações do desenvolvedor com o intuito de agilizar o processo de catalogação e minimizar o tempo dispendido no preenchimento de diversos metadados onde muitas vezes o usuário não tem o menor conhecimento de qual informação utilizar em determinados campos.

Figura 6 – Tela que solicita a localização física do objeto de aprendizagem (*wizard* para nível iniciante). Fonte autor.

O objeto de aprendizagem catalogado no exemplo necessita apenas de informações básicas a respeito de seu conteúdo que o desenvolvedor de OA deve fornecer sem maiores problemas em adquirir tais informações. Na figura 6 é necessário somente o preenchimento do campo onde estão localizados os conteúdos, as informações pertencentes ao grupo de metadados referências e informações gerais do objeto de aprendizagem são inferidas automaticamente após a análise do *upload* do conteúdo, informações como tipo do arquivo, tamanho, plataforma de operação, extensões, etc...

Também no exemplo apresentado pode ser visto que o *wizard* minimizou as etapas de preenchimento dos metadados em três etapas distintas, sendo a primeira para a identificação do objeto de aprendizagem a próxima etapa é de informações para o armazenamento e localização posterior do objeto, na terceira etapa o *wizard* identificou que este objeto de aprendizagem diz respeito ao domínio de ensino e realizou a solicitação destas informações e a ultima etapa é somente para que o usuário confirme os dados informados para o *wizard*.

A criação e montagem das etapas e telas apresentadas ao usuário é realizada de forma dinâmica e utiliza informações de domínio de ensino do OA, aplicação deste objeto, plataforma em que será utilizado, metodologia de ensino e outros tópicos sugeridos pelo projeto OBAA-MILOS.

O preenchimento das informações dos metadados, para o perfil iniciante como apresentado na figura 6 propõem um sistema com o mínimo de informações do usuário e com o preenchimento correto dos metadados para que o desenvolvedor tenha sua atenção exclusiva para a criação do conteúdo do OA. O processo de catalogação de objetos de aprendizagem é criado pelos respectivos agentes *wizards* para que atendam somente a necessidade mínima do desenvolvedor de OA levando em consideração o grau de conhecimento deste em relação aos objetos de aprendizagem. Em um perfil “expert” o *wizard* terá a intenção de extrair o maior número de informações possíveis do usuário para que os metadados personalizados e preenchidos em sua totalidade pelo desenvolvedor, diferente dos outros perfis que os *wizard’s* realizam o preenchimento automático das informações.

6. Considerações Finais

Atualmente ferramentas de *software* como o DSPACE (*www.dspace.org*) permitem realizar a catalogação de todos os metadados de um objetos de aprendizagem quando esses metadados seguem um padrão relativamente simples, com poucos metadados, como é o caso do DSPACE que usa os metadados DCMI. Porém quando essas ferramentas são adaptados para o uso com outros tipos de metadados, como a solução DSPACE-OBAA que adapta o DSPACE para operar com os metadados OBAA, isso resulta em um aplicativo complexo e de difícil operação (ver Figura 7), que exige o preenchimento de um grande volume de informações, ocasionando, assim, muitas vezes um desânimo e uma eventual desistência dos desenvolvedores ou projetistas do objeto de aprendizagem de efetuar a correta catalogação deste objeto.

Na figura 7 é mostrada a ferramenta de catalogação DSPACE com os onze grupos de metadados do OBAA [Gluz e Vicari, 2011], em destaque as etapas que o desenvolvedor de OA deve percorrer para preencher todas as informações do catalogo para que os metadados do objeto de aprendizagem estejam corretos.

Figura 7. Interface do DSPACE-OBAA. Fonte autor.

A expectativa do sistema LINNAEUS é justamente reduzir de forma significativa o volume de informações que o projetista tem que catalogar. Isso será feito através de um extensivo uso de *wizards*, tal como delineado na seção anterior, que incorporam os conhecimentos de ontologias sobre domínios de ensino e sobre aplicações educacionais (incluindo interoperação e acessibilidade de OAs) que sejam compatíveis com a ontologia de metadados OBAA [Gluz e Vicari, 2011].

O sistema LINNAEUS terá sua aplicabilidade testada em conjunto com outros subsistemas presentes na proposta OBAA-MILOS, o sistema esta disponível para acessos *online* e preenchimento de metadados suportados pela proposta OBAA. Em um próximo passo o sistema LINNAEUS será integrado com o sistema de autoria de conteúdo para ter a sua totalidade de serviços testados e validados pelo usuário final. A avaliação de sua usabilidade e aplicação será comparada a outras ferramentas de catalogação existentes, no decorrer do desenvolvimento e implementação do sistema LINNAEUS.

A arquitetura do sistema de catalogação LINNAEUS pretende viabilizar o desenvolvimento de um *software* para a catalogação de objetos de aprendizagem que possa ser juntar-se a uma solução de maior âmbito que compreenda um ambiente de autoria, recuperação e consulta de objetos de aprendizagem [Gluz e Vicari, 2010; Gluz e Vicari,

2012]. Espera-se com os resultados deste trabalho validar a aplicabilidade da proposta de catalogação para objetos de aprendizagem compatíveis com a proposta OBAA.

7. Agradecimentos

Os autores agradecem ao MCT, FINEP, FUNTTEL, CNPq e a CAPES por financiarem esta pesquisa.

Referências

BEZ, M., VICARI, R. M. SILVA, J. M. C., RIBEIRO, A., GLUZ, J. C., PASSERINO, L. M., SANTOS, E. PRIMO, T., ROSSI, L., BEHAR, P., FILHO, R., ROESLER, V. Proposta Brasileira de Metadados para Objetos de Aprendizagem Baseados em Agentes (OBAA). RENOTE. Revista Novas Tecnologias na Educação. v.8, p.1 - 10, 2010.

GLUZ, J.C. e VICARI, R. MILOS: Infraestrutura de Agentes para Suporte a Objetos de Aprendizagem OBAA. Anais do SBIE 2010, João Pessoa, 2010.

GLUZ, J. C.; VICARI, R. Uma Ontologia OWL para Metadados IEEE-LOM, Dublin-Core e OBAA. Anais do SBIE 2011, Aracaju, 2011. v. 1. p. 204-213.

GLUZ, J.C., VICARI, R e PASSERINO, L. An Agent-based Infrastructure for the Support of Learning Objects Life-Cycle. Procs. of ITS 2012, Chania, Creta, 2012. Lecture Notes in Computer Science. New York: Springer, 2012. v. 7315. p. 691-693.

IEEE Learning Technology Standards Committee (LTSC). Standard for Learning Object Metadata, IEEE Standard 1484.12.1. Nova York, Institute of Electrical and Electronics Engineers, 2002.

JENNINGS, N.; SYCARA, K.; WOOLDRIDGE, M. A Roadmap of Agent Research and Development. 1998. Em Journal Autonomous Agents and Multi-Agent Systems, Holanda.

KOERNER, L. 1999. Linnaeus: Nature and Nation. Harvard University Press, Cambridge.

VICARI, R.; GLUZ, J.; PASSERINO, L. M. ; SANTOS, E.; PRIMO, T.; ROSSI, L.; BORDIGNON, A. ; BEHAR, P.; ROESLER, V. The OBAA Proposal for Learning Objects Supported by Agents. Procs. of AAMAS 2010 – MASEIE Workshop, Toronto, 2010.

WOOLDRIDGE, M.; JENNINGS, N. Agent Theories, Architectures, and Languages: A Survey. 1994. Amsterdam, Holanda. Em Workshop on Agent Theories, Architectures and Languages. p. 1-32.

WOOLDRIDGE, M. An Introduction to MultiAgent Systems. 2002. John Wiley & Sons Ltd, paperback.

WEISS G., 1999. Learning in Multiagent Systems. In G.Weiss, editor, Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, pages 559-298. The MIT Press, Cambridge, MA.

ANEXO II - ARTIGOS PUBLICADOS

Anais do XXIV SBIE - XVIV CBIE Campinas, 25 a 29 de novembro de 2013. Uma ferramenta para fornecer apoio a catalogação de metadados de objetos de aprendizagem - LINNAEUS.

Uma ferramenta para fornecer apoio a catalogação de metadados de objetos de aprendizagem - LINNAEUS

Ederson Luiz Silveira[1], Matheus Campezatto Galão[1], João Carlos Gluz[1]

[1]PIPCA – Programa Interdisciplinar de Pós-Graduação em Computação Aplicada– Universidade do Vale dos Sinos (UNISINOS)
Av. Unisinos, 950 – 93.022-000 – Cristo Rei – São Leopoldo – RS – Brasil

esilveira1@gmail.com, matheusz.cg@gmail.com, jcgluz@unisinos.br

Abstract. *This work presents Linnaeus, a tool that supports the cataloging process of learning objects, working with the OBAA metadata standard. The Linnaeus uses an innovative technological solution to support this process of cataloging, integrating the technologies of intelligent agents and educational ontologies in its project. The project aims to provide an intelligent and proactive support, helping users without technical knowledge about learning objects or metadata standards to correctly catalog their objects. This article presents the architecture of Linnaeus and system's ontologies, showing the main features of the prototype tool and the initial results of its use.*

Resumo. *Este trabalho apresenta a ferramenta Linnaeus que apoia o processo de catalogação de objetos de aprendizagem, considerando o padrão de metadados OBAA. O Linnaeus usa uma solução tecnológica inovadora para suportar este processo de catalogação, integrando as tecnologias de agentes inteligentes e de ontologias educacionais em seu projeto. O projeto se propõe a fornecer um apoio inteligente e pró-ativo, ajudando usuários sem conhecimentos técnicos sobre metadados ou padrões de objetos de aprendizagem a catalogar de forma correta seus objetos. O artigo apresenta a arquitetura do Linnaeus e as ontologias aplicadas no sistema, mostra as principais características do protótipo da ferramenta e os resultados iniciais de sua utilização.*

1. Introdução

Objetos de Aprendizagem (OA) são apresentados como qualquer entidade, que pode ser utilizada, reutilizada ou referenciada durante o aprendizado apoiado por computador (IEEE-LTSC, 2002). Para possibilitar a reutilização, descoberta e facilitar a interoperabilidade entre os objetos foram criados padrões de metadados, tais como: IEEE LOM (IEEE-LTSC, 2002), Dublin Core (Kunze e Baker, 2007), SCORM (ADL, 2001) e OBAA (Bez et al., 2010).

Devido a necessidade contínua de atingir mais aplicações e domínios, os padrões de metadados vêm se tornando cada vez mais complexos e extensos. Consequentemente, o tempo e esforço gastos para o preenchimento dos metadados pelos projetistas de OAs têm aumentado. O preenchimento dos metadados é uma atividade importante, formando a base

dos processos de catalogação, indexação e localização dos objetos. O preenchimento manual destas informações, levando em consideração o complexo cenário de padrões de metadados, tende a gerar divergências e erros nas informações, devido a possíveis diferenças de interpretação quanto ao significado do metadado.

Neste contexto, a criação de ferramentas que auxiliem no processo de autoria de OAs, incluindo o processo de catalogação, torna-se um ponto fundamental na utilização de qualquer padrão de OA. Diversas ferramentas de autoria foram desenvolvidas com o intuito de facilitar o desenvolvimento de OAs nos diversos padrões existentes. Dentre os exemplos encontrados pode-se citar: Aprendaris.cl, Atenex, CourseLab, eXe Learning, FreeLOms, Lectora, LOMPad, Xerte e DSPACE. Várias dessas ferramentas, como Aprendaris.cl, Atenex, CourseLab, FreeLoms, LOMPad e Lectora são basicamente editores de conteúdo, não tratando das questões de catalogação do objeto. Assim, eXe Learning (BARBONE e RIFON, 2009) e Xerte (BALL e TENNEY, 2008), que combinam facilidades de autoria de conteúdo com suporte à catalogação, e DSPACE (SMITH et al., 2003), que é exclusivamente direcionada ao processo de catalogação, são as ferramentas mais relacionadas ao presente trabalho. Todas essas ferramentas disponibilizam uma interface gráfica para o preenchimento dos metadados. Apesar destas ferramentas oferecerem um mecanismo de entrada de informação organizado com vocabulário enumerado e explicações sobre a semântica do metadado, as informações ainda tem que ser manualmente preenchidas pelo autor do OA. Faltam mecanismos que auxiliem o preenchimento dos metadados de forma inteligente, ou seja, sem que o autor ou projetista do OA tenha que informar todos os dados explicitamente, este é um tópico que pode ser melhorado.

Em vista disso, este trabalho apresenta a ferramenta Linnaeus que apoia o processo de catalogação de OAs relacionado ao preenchimento dos metadados. A ferramenta auxilia as atividades de criação e edição dos metadados de OAs para domínios educacionais. Para isto o Linnaeus utiliza mecanismos para preenchimento automático dos metadados, utilizando técnicas de inferência aplicadas sobre ontologias de conteúdos educacionais e a ontologia OBAA (Gluz e Vicari, 2011).

O sistema Linnaeus opera somente na camada de metadados. O protótipo deste sistema será integrado à infraestrutura MILOS (Gluz e Vicari, 2010; Gluz et al., 2012), fornecendo parte do serviço de autoria de OA a ser disponibilizado pela infraestrutura. Assim o Linnaeus efetua um intercâmbio de informações com os demais subsistemas da infraestrutura.

A organização deste artigo é a seguinte: a Seção 2 a arquitetura do sistema e dos seus componentes, na Seção 3 é apresentada um uso de caso do sistema juntamente com um passo-a-passo e na seção 4 são apresentas as considerações finais.

2. Arquitetura do Sistema Linnaeus

O principal cenário de aplicação da ferramenta Linnaeus (Silveira e Gluz, 2012) é fornecer apoio a usuários sem conhecimentos técnicos sobre metadados ou padrões de objetos de aprendizagem e as atividades de criação e edição dos metadados de Objetos de Aprendizagem para domínios educacionais. Neste contexto, os principais requisitos levados em conta no projeto da interface do sistema são a disponibilização de uma interface de edição Web para a troca de informações com o usuário, em conjunto com o desenvolvimento de mecanismos de preenchimento e validação do conteúdo dos metadados baseado em ontologias de domínios educacionais. Os mecanismos de preenchimento e validação de metadados permitem que a interface com o usuário esteja mais próxima do domínio educacional do OA.

2.1 Infraestrutura MILOS

A arquitetura de software definida para o Linnaeus (Silveira e Gluz, 2012) deve estar alinhada e integrada com a infraestrutura de agentes MILOS (Gluz e Vicari, 2010; Gluz et al. 2012), devendo ser considerada um dos componentes da própria MILOS. A arquitetura da MILOS é dividida em três grandes níveis de abstração (ver Figura 1): a) Nível das Ontologias: responsável pela especificação dos conhecimentos que serão compartilhados entre os agentes da infraestrutura; (b) Nível de Agentes: responsável pela implementação do suporte aos requisitos de adaptabilidade, interoperabilidade e acessibilidade previstos no padrão de metadados OBAA (Bez et al., 2010); (c) Nível das Facilidades de Interface: responsável pela comunicação dos agentes da MILOS com servidores Web, ambientes virtuais, repositórios de OA, bancos de dados, serviços de diretórios e demais tipos de aplicações educacionais.

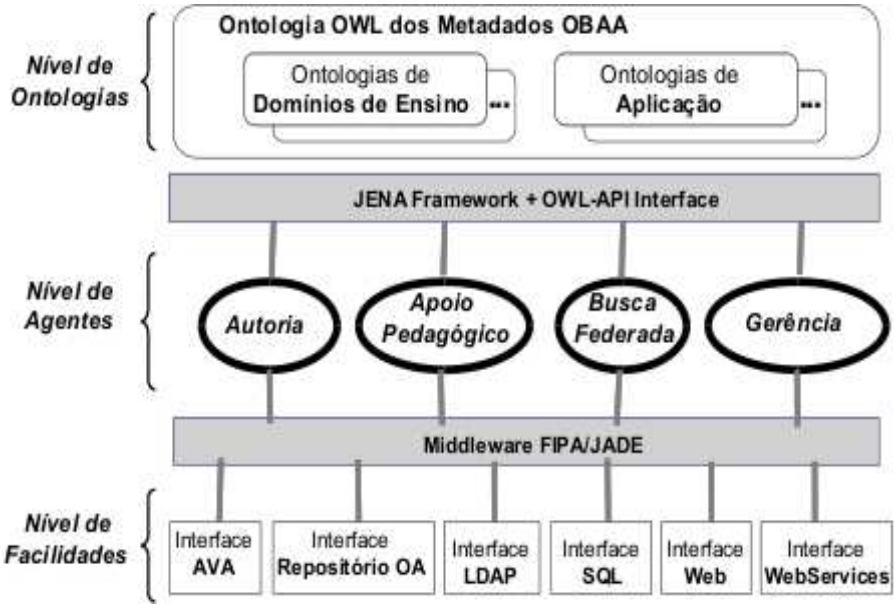


Figura 1 – Organização geral da MILOS. Fonte (Gluz e Vicari, 2010).

O Linnaeus faz parte do Sistema de Autoria da MILOS, cujo principal objetivo é prover assistência para os processos de autoria de OA, incluindo autoria de metadados e conteúdos (ver Figura 2).

Os agentes da camada de inteligência do sistema de preenchimento de metadados de OAs incorporam as principais facilidades do sistema de catalogação. Estas facilidades são acessadas diretamente por meio de uma interface Web. Por trás da operação dos agentes de preenchimento existe um conjunto de agentes em forma de Wizards de edição que são responsáveis pelo apoio a atividades de catalogação específica.

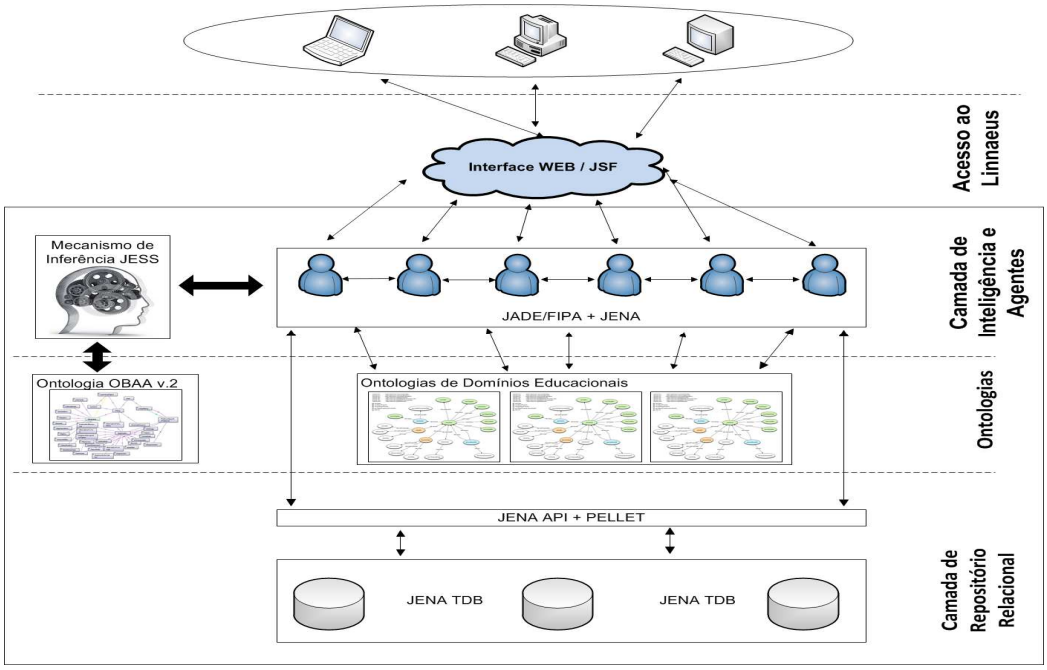


Figura 2 – Organização do sistema Linnaeus. Fonte (autor, 2013).

2.2 Agentes do Sistema Linnaeus

A principal função do Linnaeus dentro da MILOS é disponibilizar um conjunto de wizards para o apoio ao preenchimento de metadados compatível com os metadados para objetos OBAA. Porém, tendo em vista o estágio inicial de desenvolvimento da infraestrutura MILOS, o Linnaeus também se tornou um protótipo inicial para um sistema de apoio inteligente e pró-ativo, ajudando usuários sem conhecimentos técnicos sobre metadados ou padrões de objetos de aprendizagem a catalogar de forma correta seus objetos. Assim, essa ferramenta foi concebida desde o início nas três camadas da MILOS: ontologias, agentes e interface com o usuário. As operações e tarefas do Linnaeus são implementadas por meio de agente de software (Wooldridge, 2002; Weiss, 1999), usando o conceito de agentes wizards inteligentes. A criação do catalogo de metadados do sistema se dará de forma automática sem o conhecimento do usuário, o catalogo seguirá como uma de suas premissas o domínio do OA, sua aplicabilidade, plataforma de operação, para a criação do catalogo.

A modelagem do comportamento dos agentes deste sistema é baseada em workflow de regras de negócio BPM (Business Process Management) com o auxílio da ferramenta WADE (Caire et al, 2008), que é um subsistema da plataforma JADE. Seguindo esta ideia, serão apresentados o comportamento de dois importantes agentes do sistema.

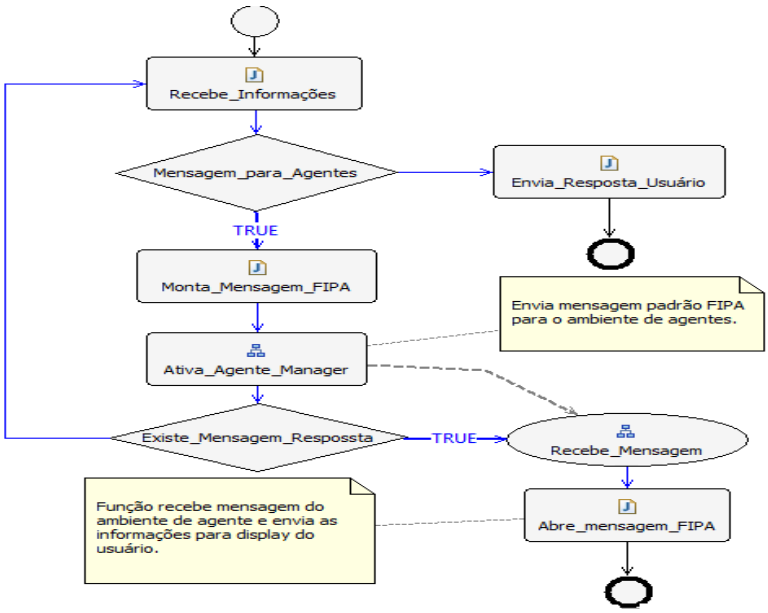


Figura 3 – Comportamento do agente responsável pela comunicação com o usuário. Fonte (autor, 2013).

O agente Gateway (figura 3) é o responsável por toda interação com o usuário. A cada solicitação ou resposta do projetista, este agente acionará o agente Manager que por sua vez acionará os demais agentes quando necessário, para que as tarefas solicitadas sejam atendidas. A troca de mensagens entre o agente Gateway e o Manager ocorre de forma que as informações fornecidas pelo usuário são convertidas para mensagens no formato FIPA, por uma classe extra adicionada ao agente.

O agente Manager (figura 4) é o organizador das informações sobre definição dos metadados, validação de valores preenchidos manualmente e também pela solicitação de informações ao usuário. Este agente é o responsável por disparar o agente que realizará a inferência dos valores dos metadados a partir das ontologias definidas em conjunto com a ontologia OBAA (Gluz e Vicari, 2011). Além dos agentes responsáveis pelo controle e organização do mecanismo de inferência, este agente também coordena os agentes responsáveis pelo acesso a base de triplas RDF, onde são armazenados os novos metadados gerados pelo mecanismo de inferência .

A geração dos metadados de saída é responsabilidade do agente Manager. Este agente também é responsável pela validação das informações fornecidas pelo projetista, no momento em que o mecanismo de inferência não consiga resolver a análise que esteja executando. As mensagens trocadas entre os agentes são, principalmente, os próprios conjuntos de metadados gerados no processo de catalogação.

Este agente é responsável pela execução correta dos demais agentes do sistema. Basicamente o agente Manager centraliza as informações e as distribui de forma correta aos demais, sem que haja conflitos de requisição e sobrecarga de tarefas a serem executadas.

O agente Manager possui suporte a mensagens provenientes de outros sistemas multiagentes, em caso do recebimento de mensagens deste tipo o sistema possui uma chamada para um agente que fara a comunicação com outros sistemas para eventuais consultas a base de metadados ou até mesmo a alteração de informações contidas nesta base.

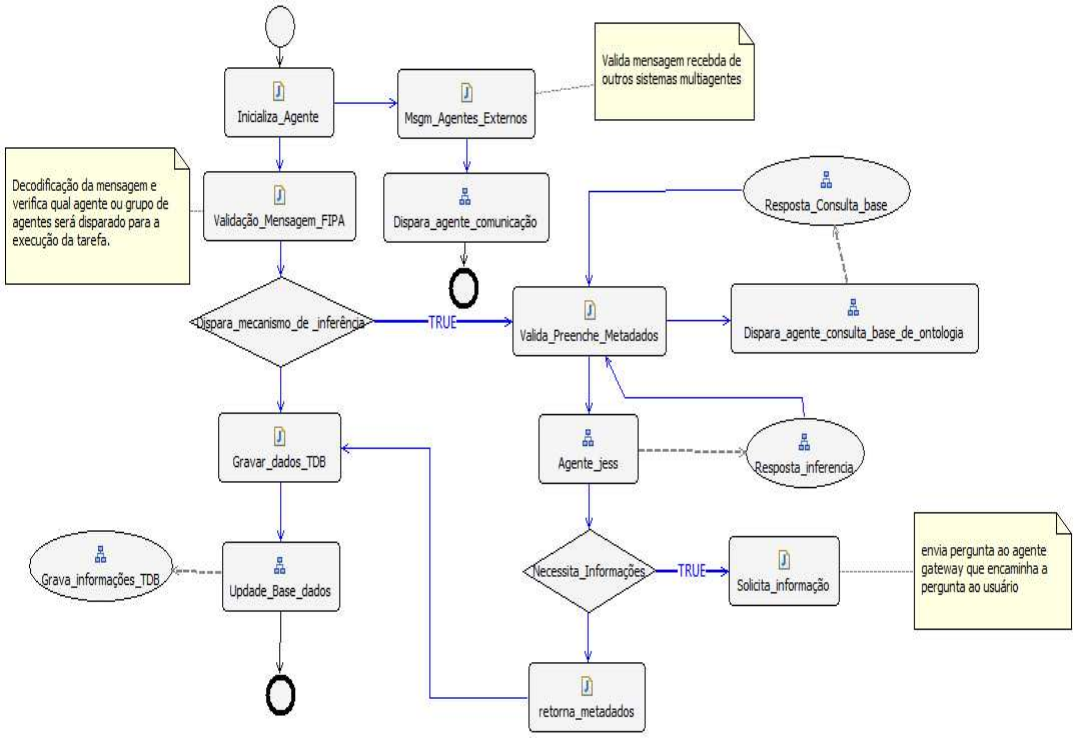


Figura 4 – Comportamento do agente responsável pelo gerenciamento e controle dos demais agentes. Fonte (autor, 2013).

3. Interface Adaptativa do Linnaeus

O protótipo do sistema Linnaeus foi projetado para ser utilizado por desenvolvedores de OA. O Linnaeus oferece distintos de interface de usuário, que se adaptam aos conhecimentos do desenvolvedor em relação aos aspectos técnicos dos OA. São considerados três níveis de conhecimentos: desenvolvedores leigos, sem conhecimentos técnicos sobre OA ou metadados, desenvolvedores com conhecimentos intermediários sobre estes temas e desenvolvedores que se consideram especialistas (experts) nestes conhecimentos. Seguindo esta caracterização, as interfaces de usuário do Linnaeus são especificadas em três casos de uso distintos (ver Figura 5), sendo o primeiro para usuários classificados como iniciantes devido ao nível de conhecimento técnico sobre OA, o segundo em nível intermediário de conhecimento técnico e o terceiro em nível expert com grande conhecimento técnico em OA.

A seguir são detalhados as principais características do caso de uso de usuários leigos, opção que leva à interface de usuário do Linnaeus que oferece maior suporte ao processo de catalogação. Será apresentado um processo passo-a-passo de catalogação de um OA utilizando o Linnaeus, mostrando a interação do sistema com o usuário através de pop-ups web que solicitam informações ao usuário sempre que o mecanismo de inferência necessitar de informações adicionais para a catalogação. As perguntas são geradas pelo agente que coordena o mecanismo de inferência, estas perguntas são geradas de uma base de perguntas pré-selecionadas para que o usuário possa respondê-las de forma imediata sem a necessidade de buscar informações e ajuda para tal questionamento. O sistema Linnaeus esta definido com dois tipos de perguntas que serão aplicadas ao desenvolvedor, a primeira pergunta é do tipo EditBox, para entrada de dados do tipo texto. O segundo tipo de pergunta é do tipo RadioBox para respostas do tipo booleanas do tipo VERDADEIRO/FALSO ou SIM/NÃO.

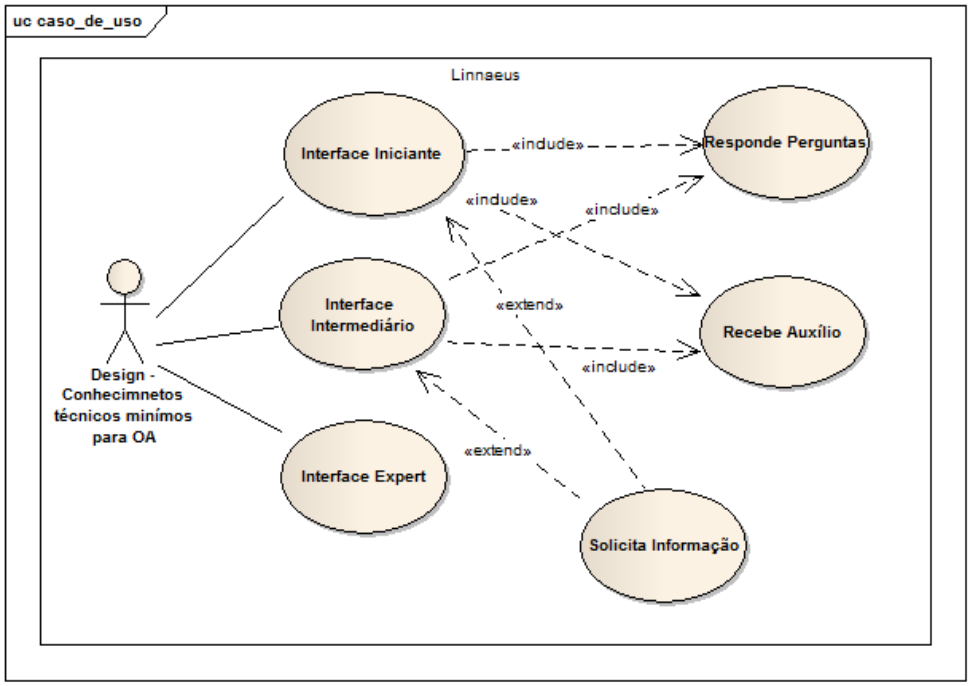


Figura 5 – Diagrama de Casos de Uso do Linnaeus. Fonte [autor, 2013].

Ao acessar o sistema o usuário encontra a área onde deve fornecer informações de identificação (figura 6) e grau de conhecimento técnico com relação a objetos de aprendizagem. Juntamente com o idioma nativo do OA que será catalogado. No momento em que o botão de login é executado os agentes do sistema juntamente com o mecanismo de inferência são inicializados e o processo de catalogação de metadados fica em modo de espera para realizar a consulta na base de ontologias para o domínio definido pelo desenvolvedor de OA.

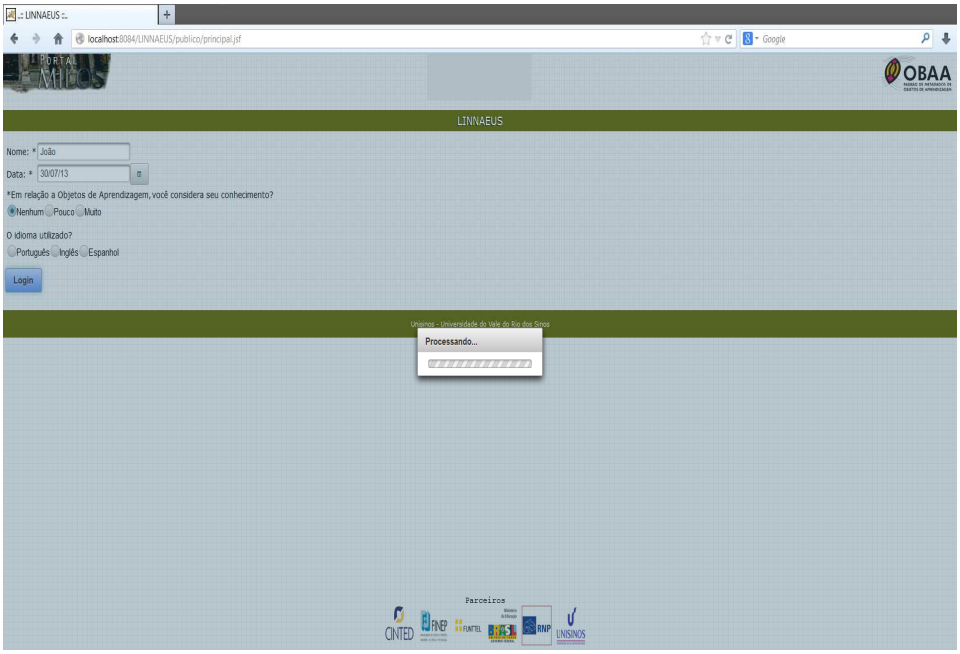


Figura 6 – Tela de informações iniciais fornecidas pelo desenvolvedor. Fonte [autor, 2013].

O próximo passo executado pelo desenvolvedor no Linnaeus é responder a solicitação do domínio do OA que será catalogado. Através de um entrada de dados do tipo EditBox (figura 7) com esta informação o sistema executa a pesquisa na base ontologias de domínios

educacionais e transmite estas informações ao mecanismo de inferência para a continuidade do processo de preenchimento de metadados. Quando o mecanismo de inferência não conseguir resolver qual informação será atribuída a determinado metadado e este tiver o tipo texto o mecanismo de inferência envia a solicitação de que precisa de informações adicionais ao seu agente de gerenciamento e este encaminha a solicitação para que o desenvolvedor de OA possa responde-la e encaminha o conteúdo desta resposta novamente para o mecanismo de inferência e o processo segue até o termino do preenchimento dos metadados.

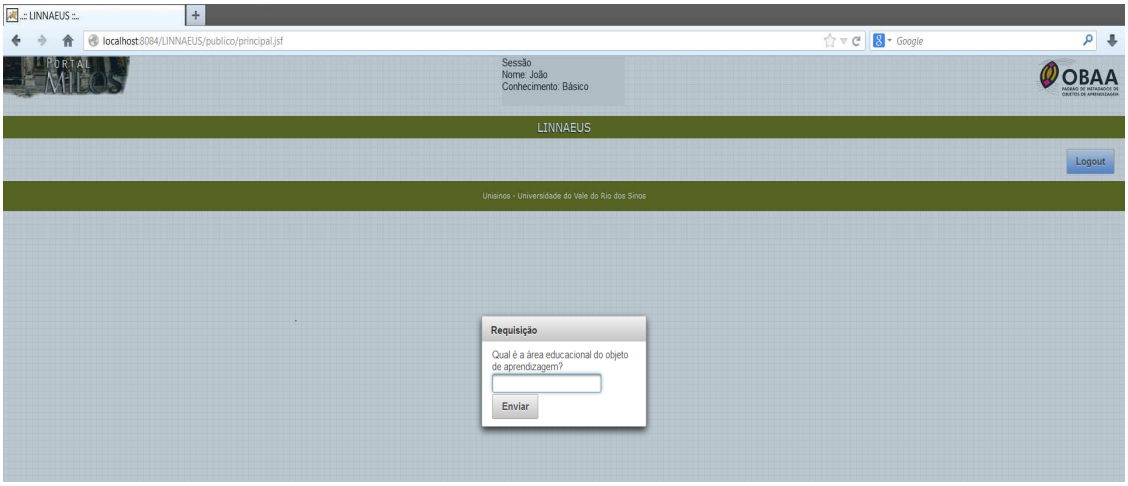


Figura 7 – Tela de entrada de dados do tipo texto. Fonte [autor, 2013].

Caso ocorra a necessidade de informações que utilizem respostas booleanas o sistema encaminha ao desenvolvedor de OA um questionamento do tipo RadioBox conforme mostrado na figura 8.

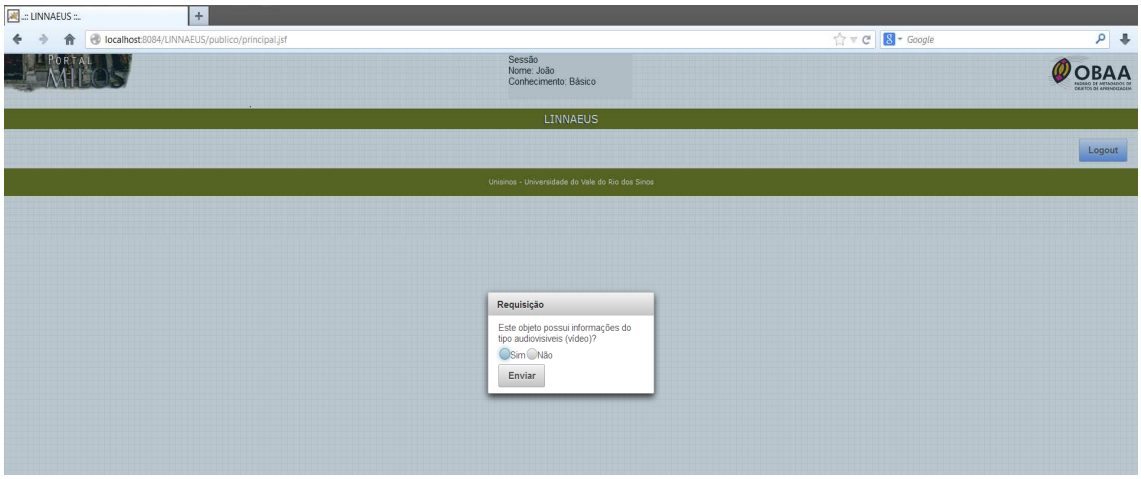


Figura 8 – Tela de entrada de dados do tipo RadioBox. Fonte (autor, 2013).

Se o Linnaeus não encontrar nenhum problema ou dificuldade para o preenchimento dos metadados, e o processo tiver percorrido todos os itens definido pelo padrão de metadados OBAA o sistema finaliza todas as tarefas e gera os metadados para serem gravados na base de dados de triplas RDF.

O próximo passo a ser executado pelo desenvolvedor de OA é a finalização do processo. O sistema apresenta ao usuário os metadados gerados (figura 9) que podem ser alterados por edição manual sem o auxílio do sistema de agentes ou o desenvolvedor poderá finalizar o processo aceitando os metadados gerados pelo sistema através do botão de conclusão da tarefa.

Ao executar o botão de conclusão o sistema dispara a gravação dos metadados na base de dados de triplas RDF e uma nova sessão é iniciada para a criação de um novo catalogo.

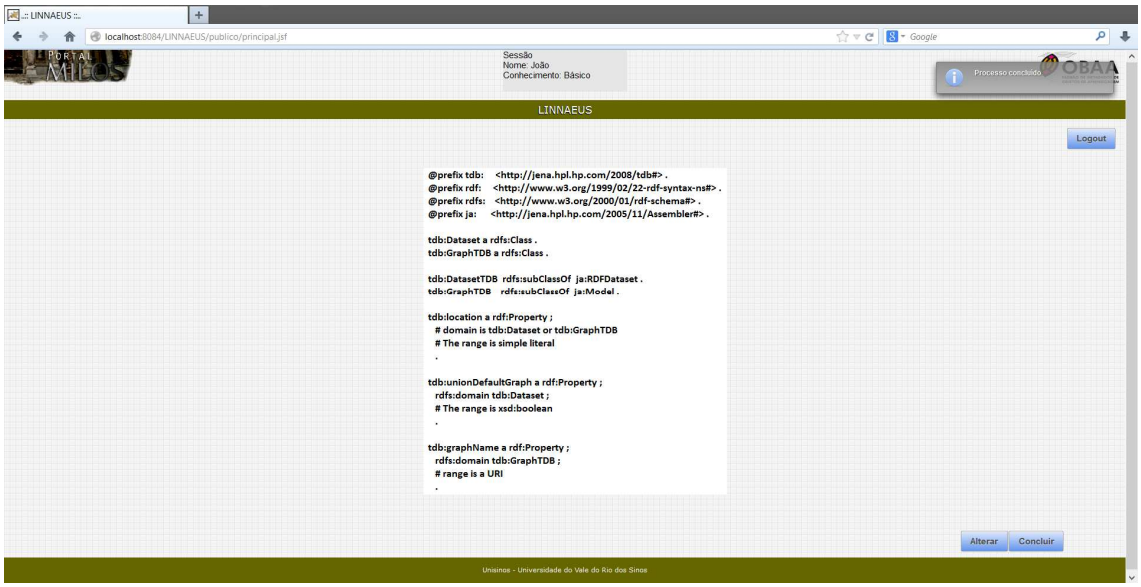


Figura 9 – Tela de apresentação dos metadados gerados pelo sistema no formato de triplas RDF. Fonte [autor, 2013].

4. Considerações Finais

Este trabalho apresentou a evolução da ferramenta Linnaeus, uma implementação da arquitetura multiagente proposta para apoiar o processo de catalogação de objetos de aprendizagem. A expectativa do sistema LINNAEUS é reduzir de forma significativa o volume de informações que o projetista tem que catalogar. Atualmente o sistema Linnaeus esta em processo de validação de suas principais funcionalidade e de aprimoramento do mecanismo de inferência.

O sistema LINNAEUS terá sua aplicabilidade testada por usuários finais no próximo mês (agosto), estando disponível para acessos online e preenchimento da avaliação de seu desempenho visto por este usuário. Espera-se com os resultados deste trabalho validar a aplicabilidade da proposta de catalogação para objetos de aprendizagem compatíveis com padrão de metadados OBAA.

Espera-se que o sistema Linnaeus seja uma contribuição para um modelo inicial para a arquitetura do subsistema de Autoria da Infraestrutura MILOS (Gluz e Vicari, 2010; Gluz et al., 2012), mostrando a viabilidade do sistema de forma prática, através do desenvolvimento de um protótipo funcional para este subsistema.

5. Agradecimentos

Os autores agradecem ao MCT, FINEP, FUNTTEL, CNPq e a CAPES por financiarem esta pesquisa.

Referências

ADL. Advanced Distributed Learning Initiative. *Sharable Content Object Reference Model (SCORM) Version 1.3: The SCORM Overview*. Alexandria: ADLnet, 2001. Disponível em: <http://www.adlnet.org>. Acesso em: out/2012.

BARBONE,V.G., RIFON,L.A. From SCORM to Common Cartridge: A step forward, 2009. *Computers & Education*, 2010.88-102.

BALL,S., TENNEY, J. Xerte – A User-Friendly Tool for Creating Accessible Learning Objects, 2008. *Lecture Notes in Computer Science* 5105, pp. 291 a 294.

BEZ, M, VICARI, R. M., SILVA, J., RIBEIRO, A., GLUZ, J. C., PASSERINO, L. , SANTOS, E., PRIMO, T., ROSSI, L., BEHAR, P., FILHO, R. Proposta Brasileira de Metadados para Objetos de Aprendizagem Baseados em Agentes (OBAA). *RENOTE*, v. 8, p. 1-10, 2010

CAIRE, G., GOTTA, D., BANZI, M. WADE: A software platform to develop mission critical applications exploiting agents and workflows. *Procs. of AAMAS'08*. Portugal: 29-36.

GLUZ, J.C. e VICARI, R. MILOS: Infraestrutura de Agentes para Suporte a Objetos de Aprendizagem OBAA. *Anais do SBIE 2010*, João Pessoa, 2010.

GLUZ, J. C.; VICARI, R. Uma Ontologia OWL para Metadados IEEE-LOM, Dublin-Core e OBAA. *Anais do SBIE 2011*, Aracaju, 2011. v. 1. p. 204-213.

GLUZ, J.C., VICARI, R e PASSERINO, L. An Agent-based Infrastructure for the Support of Learning Objects Life-Cycle. *Procs. of ITS 2012*, Chania, Creta, 2012. Lecture Notes in Computer Science. New York: Springer, 2012. v. 7315. p. 691-693.

IEEE Learning Technology Standards Committee (LTSC). *Standard for Learning Object Metadata, IEEE Standard 1484.12.1*. Nova York, Institute of Electrical and Electronics Engineers, 2002.

JENNINGS, N.; SYCARA, K.; WOOLDRIDGE, M. A Roadmap of Agent Research and Development. *Journal Autonomous Agents and Multi-Agent Systems*, Holanda, 1998.

KUNZE, J.; BAKER, T. *The Dublin Core Metadata Element Set: RFC 5013*. California: IETF, 2007.

SILVEIRA, E. L., GLUZ, J., Sistema LINNAEUS: apoio inteligente para a catalogação e edição de metadados de objetos de aprendizagem. *Anais do SBIE 2012*, Rio de Janeiro, 2012.

SMITH, M., BASS,M., MCCLELLAN, G., TANSLEY, R., BARTON, M., BRANSCHOFSKY, M., STUVE, D., WALKER, J. H. DSpace An Open Source Dynamic Digital Repository, *D-LIB Magazine*., 2003.

WOOLDRIDGE, M. *An Introduction to MultiAgent Systems*. 2002. John Wiley & Sons Ltd, paperback.

WEISS G., 1999. Learning in Multiagent Systems. In G.Weiss (Ed), *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, p. 559-298. The MIT Press, Cambridge, MA.