UNIVERSIDADE FEDERAL RURAL DO SEMIÁRIDO
UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

KLEBER JACINTO

Tohossou: Sistema Multiagente de Apoio ao Desenvolvimento de Aplicações Web Acessíveis

MOSSORÓ – RN

2012

Kleber Jacinto

Tohossou: Sistema Multiagente de Apoio ao Desenvolvimento de Aplicações Web Acessíveis

Dissertação apresentada ao Mestrado de Ciência da Computação – associação ampla entre a Universidade do Estado do Rio Grande do Norte e a Universidade Federal Rural do Semiárido, para a obtenção do título de Mestre em Ciência da Computação.

Orientadores:

Prof. Dr. Francisco Milton Mendes Neto – UFERSA.

Profa. Dra. Cicilia Raquel Maia Leite – UERN.

MOSSORÓ – RN

2012

Ficha catalográfica preparada pelo setor de classificação e catalogação da Biblioteca "Orlando Teixeira" da UFERSA

Bibliotecária: Vanessa de Oliveira Pessoa

CRB15/453

J25t Jacinto, Kleber.

Tohossou: sistema multiagente de apoio ao desenvolvimento de Aplicações Web Acessíveis. / Kleber Jacinto. -- Mossoró, 2012.

94 f.: il.

Dissertação (Mestrado em Ciência da Computação) – Universidade Federal Rural do Semi-Árido.

Orientador: Dr. Francisco Milton Mendes Neto.
Co-orientador: Dra. Cicilia Raquel Maia Leite

1.Acessibilidade. 2. Sistemas multiagentes. 3. Ontologias. I.Título.

CDD: 003.54

KLEBER JACINTO

Tohossou: Sistema Multiagente de Apoio ao Desenvolvimento de Aplicações Web Acessíveis

Dissertação apresentada ao Mestrado em Ciência da Computação para a obtenção do título de Mestre em Ciência da Computação.

APROVADA EM: ___ / ___ / _____.

BANCA EXAMINADORA

Prof. Dr. Francisco Milton Mendes Neto – UFERSA
Presidente

Prof. Dra. Cicilia Raquel Maia Leite – UERN
Primeiro Membro

Prof. Dr. Ricardo Alexsandro de Medeiros Valentim– UFRN
Segundo Membro

Prof. Dr. Danniel Cavalcante Lopes – UFERSA
Terceiro Membro

DEDICATÓRIA

À Katia Regina, meu ponto de apoio.

AGRADECIMENTOS

Aos professores do programa de Pós-graduação em Ciência da Computação UFERSA/UERN pelas centenas de horas de aulas e planejamento que fizeram os alunos crescerem como profissionais e pessoas.

Aos colegas estudantes do programa de Pós-graduação em Ciência da Computação UFERSA/UERN pelas noites insones e esforços coletivos.

Aos amigos da SUTIC, porque “uma mão lava a outra” e juntos teremos um futuro mais promissor do que enxergamos e melhor do que o que nos desejam.

A Kempes Jacinto, que dividiu comigo as atribuições de Dr Frankenstein, ao dar vida aos agentes do Tohossou.

Aos Mestres Milton e Cicília, por acatar este incorrigível mau aluno, como orientado neste trabalho, disponibilizando seu tempo, paciência e conhecimento.

À Mel, Yang, Brigite, Grandão, Nagan e Lila que num simples balançar de rabo me trazem à realidade e me mostram que a vida possui muito mais do que nossas frívolas preocupações cotidianas.

EPÍGRAFE

“Dê-me um ponto de apoio e moverei o mundo”
Arquimedes

RESUMO

Acessibilidade, no contexto da informática, refere-se ao livre acesso ao conteúdo e aos serviços, independentemente das limitações físicas e cognitivas do indivíduo, maximizando o aspecto da independência de plataforma de hardware e software do usuário. Prover este acesso, muito mais que uma questão ética, é uma questão técnica, pois as características e padrões limitantes da acessibilidade são amplamente conhecidos, mas pouco aplicados por engenheiros de software, desenvolvedores e produtores de conteúdo. Existem pelo menos três fatores que dificultam a garantia da acessibilidade neste domínio: (i) métodos tradicionais de validação de acessibilidade ocorrem a *posteriori*, exigindo refatoração e, consequentemente, retrabalho da equipe de desenvolvimento; (ii) muitas equipes trabalham hoje de forma colaborativa e geograficamente distribuídas, apoiadas em ferramentas de controle de versões de código, o que dificulta a validação de acessibilidade em partes de código de diferentes desenvolvedores; e (iii) falta de conhecimento técnico por parte dos desenvolvedores acerca das normas de acessibilidade, visto que este conhecimento não faz parte do currículo da maioria dos cursos de computação. Diante do exposto, este trabalho propõe um sistema multiagente, com uso de ontologias como base de conhecimento, que monitora o trabalho da equipe de desenvolvimento, local ou distribuída, alertando-a, em tempo real, acerca de não-conformidades quanto à acessibilidade. Além disso, os agentes são capazes de fornecer informações precisas à equipe, integrando-a, e contam ainda com a possibilidade de avaliação de múltiplos repositórios, sob múltiplas bases de conhecimento, pois é possível expandir ou incorporar novas ontologias, para mapear outros aspectos da aplicação.

Palavras-Chave: Acessibilidade, Sistemas Multiagentes, Ontologias

ABSTRACT

Accessibility, in the computing context, refers to free access to the content and services, regardless of physical and cognitive limitations of the people, maximizing the hardware and software platform independence. Provide this, more than an ethical issue, is a technical issue, because the characteristics and patterns that limits accessibility are widely known but little used by software engineers, developers and content producers. There are at least three factors that make it difficult to guarantee web accessibility: (i) traditional methods of accessibility validation occur retrospectively, requiring refactoring and rework of the development team (ii) many teams are now working collaboratively and geographically distributed, using repositories with control version tools of artifacts, hindering accessibility check, and (iii) lack of technical knowledge of developers about accessibility standards, because this knowledge is not present the curriculum of most computer science courses. Given the above, this work proposes a multi-agent system with knowledge base composed by ontologies, that monitors the artifacts of the development team, local or distributed, and warning, in real time, about non-conformities regarding accessibility. Moreover, agents are able to provide accurate information to the team, with the possibility of evaluation of multiple repositories under multiple knowledge bases and be able to incorporate new ontologies to map another aspects of the application.

Keywords: Accessibility, Multi-agent Systems, Ontologies.

LISTA DE QUADROS

QUADRO 01 – REPRESENTAÇÃO RDF/XML DE ONTOLOGIA..... 32
QUADRO 02 – TEMPLATE TEXTUAL DO AGENTE BIBLIOTECÁRIO 48
QUADRO 03 – TEMPLATE TEXTUAL DO AGENTE ANALISADOR..... 49
QUADRO 04 – TEMPLATE TEXTUAL DO AGENTE DE COMUNICAÇÃO 51
QUADRO 05 – PSEUDOCÓDIGO DE EXECUÇÃO DE AGENTE ANALISADOR 59
QUADRO 06 – PSEUDOCÓDIGO DE EXECUÇÃO DE AGENTE ANALISADOR 59
QUADRO 07 – PSEUDO-CÓDIGO DA PRIMITIVA GETPROJECT 61
QUADRO 08 – PSEUDO-CÓDIGO DA PRIMITIVA RELEASEPROJECT..... 61

LISTA DE FIGURAS

FIGURA 1 – VISÃO DO W3C SOBRE A CORRELAÇÃO ENTRE OS COMPONENTES ENVOLVIDOS NA ACESSIBILIDADE. TRADUZIDO DE: WWW.W3C.ORG. 24
FIGURA 2 - CICLO DE DESENVOLVIMENTO COM COMPATIBILIDADE À ACESSIBILIDADE. TRADUZIDO DE: WWW.W3C.ORG. 25
FIGURA 3 - VISÃO GERAL DO WAI-W3C. TRADUZIDO DE: WWW.W3C.ORG. 28
FIGURA 4 - GRAFO REPRESENTATIVO DA UMA ONTOLOGIA SIMPLES ESCRITA EM RDF. 32
FIGURA 5 - ESTRUTURA DA OWL2. FONTE: HTTP://WWW.W3.ORG/TR/OWL2-OVERVIEW/ 33
FIGURA 7 - ESTRUTURA DO REGISTRO DE UM PROJETO NO TOHOSSOU. 43
FIGURA 8 - FLUXO DE TRABALHO E AÇÕES DE CADA ATOR NO TOHOSSOU 45
FIGURA 10 - ESQUEMA BÁSICO DE AÇÃO DOS AGENTES DO TOHOSSOU 47
FIGURA 11 - ONTOLOGIA DE DOMÍNIO SOBRE TEXTOS USADA PELO TOHOSSOU. AS SETAS REPRESENTAM AS RELAÇÕES HIERÁRQUICAS ENTRE AS CLASSES 52
FIGURA 12 - CLASSES, PROPRIEDADES DE OBJETOS E DADOS DA ONTOLOGIA DE DOMÍNIO 53
FIGURA 13 - EXEMPLO DE ONTOLOGIA DE APLICAÇÃO. A CLASSE "TAG" (AS MARCAÇÕES DO HTML) É IMPLEMENTADA HERDANDO AS CARACTERÍSTICAS DA CLASSE “SENTENCES” 54
FIGURA 14 - REPRESENTAÇÃO GRÁFICA DO MODELO DE DADOS USADO PELO TOHOSSOU 56

LISTA DE SIGLAS

ABNT - Associação Brasileira de Normas Técnicas

ACL - *Agent Communications Language*

API - *Application Programming Interface*

ATAG - *Authoring Tool Accessibility Guidelines*

CDI - Comitê para Democratização da Internet

CMS - *Content Management Systems*

CVS - *Control Version System*

DAML - *DARPA Agent Markup Language*

DARPA - *Defense Advanced Research Projects Agency*

FTP – *File Transfer Protocol*

HTML - *HyperText Markup Language*

IAD - Inteligência Artificial Distribuída

ISO - *International Organization for Standardization*

ITU - *International Telecommunication Union*

KQML - *Knowledge Query and Manipulation Language*

MVC - *Model-View-Controller*

NBR – Norma Brasileira

OIL - *Ontology Inference Layer*

ONU - Organização das Nações Unidas

OO - Orientação a Objeto

OWL - *Web Ontology Language*

PHP - *Hypertext Preprocessor*

PIB - Produto Interno Bruto

PNBL - Plano Nacional para Banda Larga

RAP - *RDF API for PHP*

RDF - *Resource Description Framework*

RDFS - *RDF Schema*

RIA - *Rich Internet Applications*

SGBD – Sistema Gerenciador de Banco de Dados

SMA - Sistema Multiagente

SVN - SubVersion

UAAG - *User Agent Accessibility Guidelines*

UERN - Universidade do Estado do Rio Grande do Norte

UFERSA - Universidade Federal Rural do Semi-árido

UML - *Unified Modeling Language*

URI - *Uniform Resource Identifiers*

URL - *Uniform Resource Location*

W3C - *World Wide Web Consortium*

WAG - *Web Accessibility Guidelines*

WAI - *Web Accessibility Initiative*

WCAG - *Web Content Accessibility Guidelines*

Web - *World Wide Web*

XML - *Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO 13
1.1 CONTEXTUALIZAÇÃO 13
1.2 PROBLEMÁTICA 16
1.3 OBJETIVOS 17
1.3.1 Objetivos Específicos 18
1.4 ORGANIZAÇÃO DA DISSERTAÇÃO 18
2 REFERENCIAL TEÓRICO 20
2.1 ACESSIBILIDADE 20
2.2 ACESSIBILIDADE NA WEB 21
2.2.1 *Web Accessibility Initiative* - WAI 23
2.3 WEB SEMÂNTICA E ONTOLOGIAS 29
2.3.1 Formalização das ontologias 30
2.3.1.1 RDF 31
2.3.1.2 OWL 33
2.3.2 Aplicações em Engenharia de Software e Acessibilidade 34
2.4 AGENTES INTELIGENTES 35
2.4.1 Características e tipologia dos agentes 36
2.4.2 Sistemas Multiagente 38
3 TOHOSSOU 40
3.1 CARACTERIZAÇÃO 40
3.2 CENÁRIO DE USO 42
3.3 COMUNIDADE DE AGENTES DO TOHOSSOU 46
3.3.1 Agente Bibliotecário 47
3.3.2 Agente Analisador 49
3.3.3 Agente de Comunicação 50
3.4 BASES DE CONHECIMENTO 51
3.5 BASES DE DADOS 55
3.6 TECNOLOGIAS E TÉCNICAS UTILIZADAS E DETALHES DA IMPLEMENTAÇÃO 57
3.6.1 *Blackboard* e comunicação entre agentes 60
4 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS 62
5 REFERÊNCIAS 64
6 APÊNDICES 73
DOCUMENTO DE REQUISITOS 74
DIAGRAMA DE CASOS DE USO DO USUÁRIO 88
DIAGRAMA DE CASOS DE USO DO CONVIDADO 88
DIAGRAMA DE ATIVIDADES DE REGISTRO DE USUÁRIO 88
DIAGRAMA DE ATIVIDADES DO AGENTE BIBLIOTECÁRIO 89
DIAGRAMA DE ATIVIDADES DO AGENTE ANALISADOR 89
DIAGRAMA DE ATIVIDADES DO AGENTE DE COMUNICAÇÃO 90
MODELOS DE CONSULTAS SPARSQL 91

1 Introdução

1.1 Contextualização

A Internet é reconhecida como fonte de informação, meio de comunicação, ferramenta para agregação e melhoria da relação entre as pessoas, ambiente de aprendizado, diversão e mudança de cultura. Vêm se tornando onipresente, seja nos grandes centros, seja nos recantos de países em desenvolvimento (ITU, 2010). Este fato acentua-se quando se percebe que tecnologias como o 3G (MOHANTY, 2006) e dispositivos móveis (DIXIT *et al.*, 2011), transmissão via satélite, fibras óticas, redes sem fio em locais públicos e privados (GUNASEKARAN; HARMANTZIS, 2008), ambientes como telecentros (SALDANHA; EL HAJJI, 2011) ou *Lan Houses* (MORAES, 2009) são cada vez mais comuns.

Segundo dados recentes da *International Telecommunication Union* (ITU), cerca de um terço dos 6,8 bilhões de habitantes da Terra acessam a Internet, seja de casa, por dispositivos móveis ou de pontos de acesso. Embora ainda haja grandes distorções entre os países desenvolvidos e em desenvolvimento (apenas 15,8% dos lares destes últimos possuem acesso à Internet, em contraste a 65,6% dos países desenvolvidos), em maior ou menor escala todos os países têm passado por um forte crescimento na difusão da Internet (ITU, 2010).

A Web tornou-se não só uma forma ágil de contato entre as pessoas e dispersão de conhecimento, mas o canal primário de comunicação entre o mundo corporativo/governamental e a população, além de uma nova plataforma para o desenvolvimento de aplicações. A Web deixou de ser apenas um ambiente de documentos interconectados para tornar-se um espaço de publicação de software que realiza tarefas diversas, de jogos aos gestores de contas e transações bancárias (PUDER, 2004).

Tim O'Reilly, fundador da O'Reilly Media e um dos precursores do termo Web 2.0, afirma que esta nova Internet, que fornece uma plataforma para aplicações ricas (*Rich Internet Applications* - RIA), não é uma tecnologia, um padrão estático, um nicho de mercado, mas sim atitudes, conceitos e princípios que dependem cada vez mais da interação das pessoas. Ou seja, ela depende de que todos possam colaborar, interagir e acessar conhecimento (OREILLY, 2007).

Contudo, a universalidade não deve ser confundida com homogeneidade de acesso. Mesmo em regiões onde a Internet é amplamente usada, parcelas da população continuam a

ter o direito ao acesso negado, devido a características da forma e do conteúdo. Esta negação é comumente denominada de exclusão. A exclusão pode ocorrer essencialmente de cinco formas distintas, a saber: i) ergonomia inapropriada; ii) exclusão econômica; iii) exclusão social; iv) exclusão conceitual; e v) exclusão cultural (BENYON *et al.*, 2005).

A primeira refere-se exclusão devido à ergonomia inapropriada, vinculada ao mundo físico e a relação entre as pessoas e os equipamentos usados para acessar o conteúdo da Internet. Teclados pequenos, mouses sensíveis demais ou pouco sensíveis, o fenômeno das telas sensíveis ao toque, ou telas cada vez menores, são apenas alguns exemplos da questão da ergonomia (MCGOOKIN *et al.*, 2008).

A exclusão econômica e a exclusão social são dois aspectos intimamente relacionados, cuja existência é facilmente verificável pelos dados do ITU (ITU, 2010). Condições econômico-sociais desfavoráveis inibem a aquisição de computadores ou outros aparatos tecnológicos que permitem o acesso ou inibem a própria aquisição do canal de conectividade, como ocorre em países menos desenvolvidos. Por outro lado, em alguns países a população possui a tecnologia apropriada, mas não a liberdade de usá-la, como em alguns países comunistas ou islâmicos. Em países como o Brasil, apesar da tentativa de criação de políticas públicas, como o Plano Nacional para Banda Larga (PNBL), coordenado pelo Ministério das Comunicações (CGPID, 2010), ainda depende-se muito de iniciativas da população ou do setor privado para que se consiga permear as barreiras tanto sociais quanto econômicas. Entidades da sociedade civil organizada, como o Comitê para Democratização da Internet (CDI), são exemplos reconhecidos internacionalmente, mas que não possuem a articulação e a estrutura de um governo para alterar condições adversas históricas (VALLEJO, 2009).

As exclusões culturais e conceituais completam o conjunto de formas mais comuns de exclusão. A cultural incorpora aspectos de linguagem, uso de elementos de comunicação pouco usuais, e até mesmo uso de conjunto de caracteres ou palavras inapropriados. A conceitual abrange não só diagramação, organização, conjunto de cores, mas também aspectos técnicos como uso ou não de animações, uso de programas sob forma de *scripts* executáveis no navegador, plataformas específicas de desenvolvimento, compatibilidade de navegadores, tecnologias multimídia, dentre outros (BENYON *et al.*, 2005).

Estas duas últimas formas de exclusão são as mais correlacionadas com os problemas de acessibilidade. Acessibilidade é a condição de um dado serviço, neste caso o conteúdo de uma página ou serviço na Internet, estar pronto a atender às demandas de pessoas com limitações físicas, sensoriais ou cognitivas. A definição legal usada no Brasil, criada através da lei 10.098 de Dezembro de 2000 (BRASIL, 2000), para acessibilidade é:

> Possibilidade e condição de alcance para utilização, com segurança e autonomia, dos espaços, mobiliários e equipamentos urbanos, das edificações, dos transportes e dos sistemas e meios de comunicação, por pessoa portadora de deficiência ou com mobilidade reduzida

Observa-se que, contrariamente ao senso comum, neste grupo incluem-se crianças, idosos, analfabetos ou alfabetizados funcionais (que cursaram os anos iniciais da escola, mas não conseguem exercer a compreensão do texto escrito), portadores de doenças degenerativas do sistema motor, e não somente pessoas ditas deficientes. O conceito de acessibilidade nasceu para designar a condição de uso de espaços físicos, onde as pessoas transitam e realizam as suas atividades. A transposição deste conceito para o mundo virtual ocorreu naturalmente. Mesmo a lei brasileira, costumeiramente considerada antiquada em alguns aspectos, já previa a acessibilidade aos meios de comunicação (BRASIL, 2000).

São também estes dois aspectos os que mais dependem dos profissionais que constroem as aplicações para Web, ou mesmo para *desktop*, sejam desenvolvedores (programadores) ou projetistas (engenheiros e arquitetos de software). Compartilhando deste pensamento, o *World Wide Web Consortium* (W3C) constituiu comissões para estudar os fatores que levam à falta de acessibilidade e com isso surgiu a *Web Accessibility Initiative* (WAI[1]) que, de forma simplificada, trata de estabelecer os critérios mínimos para que ferramentas, conteúdo e diagramação de páginas Web estejam preparados para atender às demandas de acessibilidade (W3C, 2006).

Contudo, apesar de claramente definidos, estes critérios costumam ser esquecidos por diversos fatores, inclusive porque a formação educacional não costuma preparar os profissionais para este tipo de requisitos de projeto (LUDI, 2007).

Assim, prover ferramentas que sejam capazes de orientar o desenvolvimento de aplicações e conteúdo para a Web, desde suas etapas de projeto até a execução, de modo a colaborar com a criação de material compatível com os critérios de acessibilidade, não só é politicamente e socialmente justo, como possibilita minimizar a lacuna conceitual que os profissionais já formados, e em formação, possuem diante das diretivas de acessibilidade.

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

1.2 Problemática

O termo acessibilidade inicialmente foi usado para representar o acesso físico de pessoas deficientes a serviços e ambientes, mas passou a ser usado para representar a necessidade de acesso universal à informação disponível na Internet. No contexto da Web, refere-se à disposição do conteúdo para todas as pessoas, independentemente de suas capacidades físicas e cognitivas. Para que isso ocorra, há técnicas e padrões definidos, mas estes aspectos técnicos não são inteiramente aplicados pelos engenheiros de software, desenvolvedores, *designers* e produtores de conteúdo (SPELTA, 2009).

Alguns fatores são apontados como origem desta falta de aplicação, como a falta de conhecimento técnico (visto que o ensino formal e o mercado não têm acessibilidade como um requisito essencial), a escassez de ferramentas que auxiliem na garantia da acessibilidade sem retrabalho por parte da equipe e a distribuição das equipes de desenvolvimento, que dificulta a verificação do código (LUDI, 2007). Ter a acessibilidade como uma característica do produto em construção não somente é uma questão ética e social, mas também, e principalmente, uma questão técnica, que impacta diretamente no sucesso de serviços e na qualidade do atendimento a clientes destes serviços (BRUNET *et al.*, 2005).

Dados da Organização das Nações Unidas (ONU) relatam que cerca de 10% da população humana, cerca de 650 milhões de pessoas, possui algum tipo de deficiência física ou cognitiva, seja decorrente de condições genéticas ou incidentes, seja pela idade ou doenças adquiridas (UNITED NATIONS, 2010). Muitas destas pessoas ainda estão no mercado de trabalho, ainda estudam e capacitam-se, são consumidores, e são potenciais clientes para serviços *on-line* das mais diversas naturezas. Em decorrência de suas condições, usar serviços ofertados pela Web costumeiramente é mais simples do que usar os mesmos serviços em locais físicos, que demandam uso de transporte público, equipamentos de acessibilidade (rampas, banheiros adaptados, pisos táteis), auxílio de terceiros e, por vezes, constrangimentos (RITCHIE; BLANCK, 2003).

Com a maior oferta de serviços nas nuvens (*cloud computing*) (ARMBRUST *et al.*, 2010), torná-los acessíveis a todos é um desafio para o qual a união de técnicas, padrões e profissionais capacitados poderá gerar resultados satisfatórios. Como parte da solução pode-se apontar o uso de ferramentas que se integrem a equipes de desenvolvimento, apoiando os profissionais, diminuindo seus esforços, minimizando a necessidade de aporte de novos conhecimentos e responsabilidades, e auxiliando em tornar os produtos acessíveis.

As atividades envolvidas no processo de criação de software ocorrem em volume tal, que, mesmo com uma equipe disciplinada e qualificada, torna-se complexo lidar com documentos distintos (de requisitos, estudo de viabilidade, visão do projeto, entre outros), linguagens distintas (inclusive diversos diagramas em *Unified Modeling Language* - UML) e conhecimentos difusos e por vezes conflitantes dos projetistas e *stakeholders*. Some-se a isto a facilidade de comunicação atual, que permite que colaboradores distribuídos geograficamente, podendo estar em países ou continentes diferentes, trabalhem em um mesmo projeto (BRADNER *et al.*, 2003; CASEY; RICHARDSON, 2009).

Acrescentar atividades relativas à garantia da acessibilidade em um ambiente por si só tão complexo não é uma tarefa trivial. As ferramentas de validação de conteúdos acessíveis tradicionais possuem muitas limitações. Uma delas é permitir apenas a validação do produto final e não dos artefatos intermediários, gerando retrabalho para a equipe de desenvolvimento. Outra limitação consiste na dificuldade de tratar conteúdos dinâmicos, o que é incompatível com a velocidade imposta pelo avanço tecnológico e pelas demandas de mercado por aplicações mais ágeis (LUQUE CENTENO *et al.*, 2006).

Dispor de uma ferramenta que atue com a equipe de desenvolvimento durante as etapas de construção, e não apenas sobre o produto final, permite um melhor uso dos recursos humanos e financeiros do projeto (LANUBILE *et al.*, 2010). Nas ferramentas tradicionais de validação de acessibilidade, o produto final é avaliado página a página e, em alguns casos, trabalhar para excluir a não conformidade pode ser tão custoso quanto o projeto inicial (TANAKA; DA ROCHA, 2011). A escolha errada de algumas técnicas ou tecnologias logo no início ou nos primeiros ciclos de desenvolvimento pode levar à necessidade de reavaliação de todo o projeto. Realizar a avaliação da acessibilidade de forma contínua durante projeto, e não apenas no final, pode ser essencial para o sucesso do projeto.

1.3 Objetivos

Considerada toda a problemática exposta, objetivou-se nesta dissertação o desenvolvimento de uma ferramenta de Software, chamada de Tohossou (nome alusivo a uma deidade africana, protetora das pessoas deficientes) capaz de apoiar o desenvolvimento de aplicações Web com características de acessibilidade.

Neste contexto, foi projetado e desenvolvido uma aplicação nas nuvens, constituída de um sistema multiagente capaz de auxiliar na análise do cumprimento dos requisitos de acessibilidade. Os agentes apontam não-conformidades de forma on-line e possibilitam que as correções sejam feitas mais rapidamente, durante as etapas de desenvolvimento, por meio de uma base de conhecimento baseada em ontologias, passível de expansão e evolução, e da emissão de mensagens aos desenvolvedores e *stakeholders*.

1.3.1 Objetivos Específicos

- Realizar um levantamento bibliográfico que aponte as melhores técnicas e práticas pra a construção da ferramenta proposta;
- Construir um documento que permita a futuros leitores compreender a motivação, as técnicas e os resultados decorrentes deste estudo;
- Criar uma estrutura de ontologias que permita o mapeamento de características de acessibilidade;
- Criar uma estrutura de ontologias que permita a expansão da capacidade de avaliação dos agentes para outras áreas ou outros aspectos de acessibilidade;
- Construção de um sistema multiagentes preparado para interagir com usuários e sistemas através da Internet;
- Utilizar, dentro do possível, ferramentas de software livre para cumprir todas as etapas do processo de construção da dissertação e seus artefatos;

1.4 Organização da Dissertação

O Capítulo 2 apresenta do referencial teórico necessário para a compreensão do trabalho, sendo composto por um conjunto de citações da literatura que contextualizam e explicitam os conceitos que nortearam o desenvolvimento e execução do trabalho. São discutidos: a acessibilidade, de seus aspectos legais e éticos aos aspectos técnicos ligados à

sua aplicação na Web; Web Semântica e Ontologias bem como as formas de representação das mesmas; os agentes inteligentes e os sistemas multiagentes.

O Capítulo 3 apresenta o produto primário deste trabalho, o Tohossou, uma ferramenta de apoio à produção de conteúdo acessível para pessoas com deficiências. Para tanto, apresenta os conceitos envolvidos, o cenário de aplicação dos mesmos, detalhes construtivos e organizacionais da comunidade de agentes que constitui a ferramenta, as ontologias utilizadas como base de conhecimento dos agentes bem como as tecnologias e detalhes da implementação.

Por fim, o Capítulo 4 aborda as considerações finais e sugestões para a continuidade das ideias aqui apresentadas.

2 Referencial Teórico

Construir uma ferramenta computacional que permita gerar melhorias na qualidade de vida das pessoas, seja diretamente, seja pela produção de conteúdo mais adequado, passa pelo entendimento da problemática e dos aspectos éticos e legais envolvidos na geração da acessibilidade. Este conhecimento é tão ou mais importante para a execução que os aspectos técnicos e operacionais das ferramentas geradas.

2.1 Acessibilidade

Tornar direitos e deveres iguais para todos os homens e mulheres tem sido uma tarefa de todas as nações democráticas do mundo. A acessibilidade é uma das facetas deste esforço.

A acessibilidade não é um termo novo, mas que tem tomado formatos diferentes no decorrer da história. Ela pode ser definida como a condição de permitir que o uso de bens, serviços e informações, bem como o direito de ir e vir, seja expandido a todas as pessoas, independentemente de suas limitações físicas, sensitivas e cognitivas, de natureza permanente ou transitória (BERNARDES *et al.*, 2009).

Com as aberturas políticas no Velho Mundo, o fim da Guerra Fria e o fim das ditaduras no Novo Mundo, a sociedade pôde preocupar-se de forma mais eficiente com problemas como a longevidade das populações e o acesso à saúde, à educação, aos serviços públicos e aos espaços públicos. Idosos e deficientes, mais do que nunca, passaram a ser vistos pela sociedade e os espaços para estas pessoas passaram a ser reivindicados com mais força e constância (THOMAS; ELY, 1996).

No Brasil, a Constituição Federal, promulgada em 1988, possui um conjunto de princípios que visam garantir a implementação da acessibilidade. Os princípios vão desde a igualdade (art. 5º, inc. I), do acesso, permanência e atendimento especializado (art. 206, inc. I e art. 208, inc. III), da habilitação e reabilitação (art. 203, inc. IV) até a garantia da eliminação das barreiras arquitetônicas (§2º, do art. 227 e art. 244). Os objetivos fundamentais do Estado Federal Brasileiro, ou seja, a cidadania (art. 1º, inc. II), a dignidade da pessoa humana (art. 1º, inc. III) e os valores sociais do trabalho (art. 1º, inc. III), determinam que todas as decisões

judiciais, as decisões administrativas e a produção legislativa sigam estes princípios essenciais (BRASIL, 1998).

Em complemento à Carta Magna, a acessibilidade tornou-se parte de políticas públicas brasileiras com o advento do Decreto Nº 5.296, de 2 de Dezembro de 2004, e as Leis 10.048 (8 de novembro de 2000) e 10.098 (19 de dezembro de 2000), todas estabelecendo normas gerais e critérios básicos para a promoção da acessibilidade e prioridade no atendimento das pessoas portadoras de deficiência ou com mobilidade reduzida.

Em complemento, a própria sociedade vem criando normas complementares, como normas ISO (*International Organization for Standardization*). No Brasil, a Associação Brasileira de Normas Técnicas (ABNT) tem criado normas, ou apoiado as normas ISO, em diversas áreas, como as NBR 9050 (Acessibilidade a Edificações, Mobiliário, Espaços e Equipamentos Urbanos), NBR 14970-1 (Acessibilidade em Veículos Automotores-Requisitos de Dirigibilidade) ou, já com preocupação para o acesso a informações, a NBR 15250 (Acessibilidade em caixa de auto-atendimento bancário) (CORDE, 2004).

Com a popularização da Internet, a acessibilidade ganha um outro alcance: o de tornar os conteúdos acessíveis, eliminando-se, assim como no mundo físico, as barreiras do mundo virtual.

2.2 Acessibilidade na Web

Dados recentes apresentam o crescimento do número de usuários da Internet em todo o mundo, que dobrou de 2005 para 2010, e também a divergência nestes dados entre países ricos e pobres. Enquanto no continente africano apenas 9,6% da população acessa à rede, na Europa este número chega a 65%. Enquanto nos países em desenvolvimento o acesso à saúde e à alimentação ainda não é universal, países como Finlândia, Espanha e Estônia incorporaram o acesso à Internet entre os direitos essenciais dos cidadãos, ao lado de moradia e educação (ITU, 2010).

Estas diferenças vêm sendo discutidas sob o tema inclusão digital e têm maior força em países em desenvolvimento, como o Brasil, onde uma boa parcela da população não reúne condições financeiras para sanar suas necessidades essenciais, inclusive sua necessidade de

acesso à informação. O portal da Inclusão Digital do Governo Federal Brasileiro[2] fornece informações relevantes sobre as políticas de governo e diversas ações do terceiro setor (MATTOS; CHAGAS, 2008).

Outro aspecto são as limitações tecnológicas, como velocidade de conexão, espaço de armazenamento, poder de processamento, versão de software (do sistema operacional ou navegador), plataforma de hardware (*desktops*, celular, dispositivos móveis, etc.), entre outras limitações de tecnologia. Mais uma vez, este problema é especialmente relevante em países com baixo Produto Interno Bruto (PIB), onde as tecnologias demoram a popularizar-se, e faz-se necessário considerar o potencial dano ou perda de acesso se o desenvolvedor de aplicações utilizar-se sempre das tecnologias mais recentes (SPELTA, 2009). A importação torna-se a forma mais comum de aquisição de bens de tecnologia, mas impostos sobre produção e importação são constantemente apontados como vilões, e impedem a massificação do acesso a eletrônicos (BATISTA, 2010).

Outra questão a ser discutida é a acessibilidade quanto à forma como o conteúdo é disposto. Este ponto é um dos mais relevantes para o desenvolvedor de software, engenheiros e outros profissionais que planejam e executam portais e serviços baseados na Web. Mas, para atender às demandas de acessibilidade, com precisão técnica, escolha correta de plataforma e *design* adequado, faz-se necessária a existência de ferramentas ou pelo menos padrões que determinem características técnicas que permitam o acesso ao conteúdo (SPELTA, 2009).

As primeiras iniciativas de se criar padrões para a acessibilidade na Internet foram estabelecidas pelos governos Canadense, Americano e Australiano em torno do ano de 1997. Em 1998, entrou em vigor, nos Estados Unidos, a *Section 508*, uma política pública, apoiada por um conjunto de leis e normas, determinando que a tecnologia eletrônica e de informação dos órgãos federais daquele país fossem acessíveis às pessoas com necessidades especiais (JAEGER, 2006).

Contudo, até o ano de 1999, estes padrões não estavam claramente definidos e válidos para os desenvolvedores. Neste momento, o W3C, através de grupos de trabalho, definiu a *Web Accessibility Iniciative* (WAI). A partir desta iniciativa, foram determinadas as causas mais frequentes da falta de acessibilidade em muitas páginas da Web (CONFORTO; SANTAROSA, 2002). Em maio de 1999, foi lançada a *Web Content Accessibility Guidelines* 1.0 (WCAG 1.0), que rapidamente tornou-se referência mundial em termos de acessibilidade

[2] http://www.inclusaodigital.gov.br/

na Web. A versão atual é a 2.0, lançada em 11 de dezembro de 2008 (REID; SNOW-WEAVER, 2008).

Apesar do W3C preocupar-se com a criação de padrões para acessibilidade, ainda há muita resistência ao uso destas técnicas, que minimizam os impactos da falta de acessibilidade. Como consequência, tem-se milhares de páginas na Internet que não podem ser lidas por idosos, deficientes visuais, disléxicos ou daltônicos, por exemplo. Com a convergência de serviços do mundo real para o meio digital (como no comércio eletrônico, acesso a serviços bancários, prestação de contas do Imposto de renda, no caso do Brasil, etc.), estas limitações tornam-se mais evidentes e originam problemas severos, como obrigar a um paraplégico a deslocar-se fisicamente até uma agência bancária, quando o serviço poderia ser realizado pela Internet (RAMAN, 2008).

2.2.1 *Web Accessibility Initiative* - WAI

Os grupos de trabalho que compõem a WAI determinaram que a acessibilidade à Web não pode ser definida de forma pontual, dado que depende da existência e interação entre diversos componentes. Dentre estes, citam-se: i) o conteúdo da página, seja na forma escrita, seja através de recursos multimídia; ii) o navegador ou outro dispositivo de software que interpreta e apresenta o conteúdo ao usuário; iii) as tecnologias assistivas, que tratam-se de dispositivos, muitos de hardware (como teclados e mouses adaptados) e alguns de software (como os leitores de tela), que auxiliam a navegação no conteúdo; iv) os desenvolvedores (diagramadores, programadores, engenheiros de software, *stakeholders*, especialistas, inclusive portadores de deficiências, na área de conhecimento do desenvolvimento da página ou aplicação) e todos os que colaboram na criação do conteúdo; v) as ferramentas de autoria e demais ferramentas utilizadas na etapas de construção de um site ou serviço na Web; e vi) as ferramentas de avaliação de compatibilidade do produto final com padrões pré-estabelecidos (CHISHOLM, W. A.; HENRY, 2005).

De forma simplificada, são componentes técnicos (padrões, conteúdo e ferramentas) e humanos (produtores do conteúdo, criadores das ferramentas e os usuários finais) (CHISHOLM, W. A.; HENRY, 2005).

A correlação entre estes componentes pode ser visualizada na Figura 1. Os desenvolvedores devem utilizar ferramentas de autoria apropriadas para construir seu

conteúdo e validar seus artefatos, obtendo assim um conteúdo potencialmente acessível. Mas, para que o usuário possa realmente ter plena percepção deste conteúdo, os navegadores e ferramentas assistivas devem intermediar o acesso.

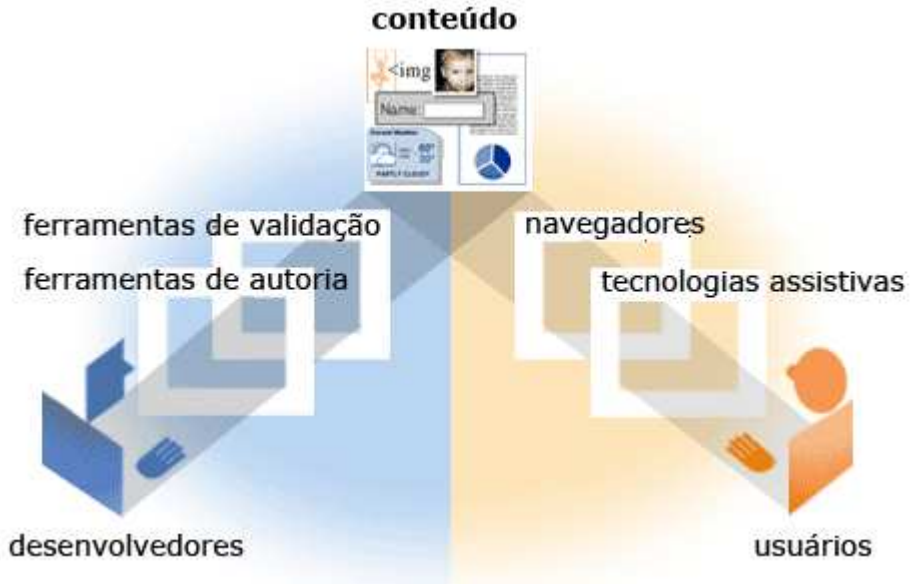


Figura 1 – Visão do W3C sobre a correlação entre os componentes envolvidos na acessibilidade. Traduzido de: www.w3c.org.

Essa visão prevê que o ciclo de produção do conteúdo deve passar pela validação em plataformas distintas, com navegadores e tecnologias assistivas diversas, conforme apresentado na Figura 2. Este ciclo deveria ser executado durante o processo de construção do conteúdo ou aplicação, mas é pouco executado na prática. A eficiência do ciclo depende de alguns fatores, como a compatibilidade das ferramentas de autoria, das tecnologias assistivas e dos navegadores com as características de acessibilidade que se pretende alcançar. Depende também da facilidade de uso das ferramentas de autoria, pois, caso contrário, não haverá adesão dos desenvolvedores, e da realização de *feedback* entre usuários e desenvolvedores. A multiplicidade de plataformas e caminhos ao qual o usuário pode chegar ao conteúdo onera excessivamente os testes.

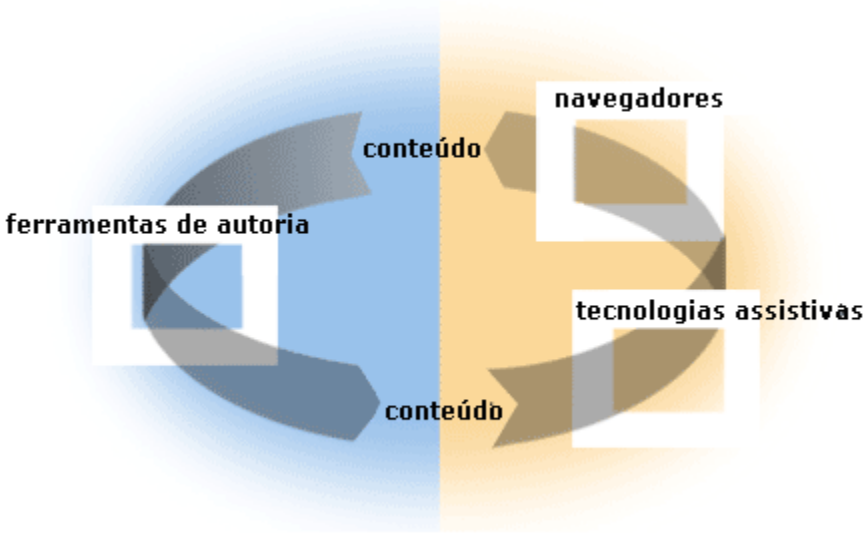


Figura 2 - Ciclo de desenvolvimento com compatibilidade à acessibilidade. Traduzido de: www.w3c.org.

Se um ou mais componentes falham ou se o ciclo não é promovido de forma adequada, a possibilidade do conteúdo não atender aos requisitos de acessibilidade é alta. Isso leva ao retrabalho das equipes de produção e ao descontentamento do público alvo.

No meio acadêmico, o ciclo de construção de software seguindo o modelo cascata é desaconselhado desde a década de 90, mas ainda é muito comum seu uso dentre os desenvolvedores, apesar dos diversos riscos envolvidos especialmente ao fato de que é comum chegar-se a um produto final que não atende à especificação de requisitos (SOARES, 2004). Pela mesma linha de raciocínio, o ciclo proposto pelo WAI costumeiramente é ignorado, e as validações não ocorrem durante o desenvolvimento, gerando produtos inadequados quanto à acessibilidade. Esta deficiência pode ser tratada pelo uso da ferramenta proposta, visto que o Torossou atua sobre os artefatos que estão sendo construídos, realizando o ciclo iterativo incremental sugerido pela WAI/W3C, e apontado pela literatura como mais eficaz que o tradicional modelo em cascata. Os agentes, atuando de forma autônoma e através da interação com os desenvolvedores, leva-os a realizar o ciclo incremental, melhorando os artefatos durante a construção dos mesmos, e não quando finalizados.

Para delinear a dinâmica proposta, o WAI e seus comitês criaram os chamados Guias para Acessibilidade na Web (*Web Accessibility Guidelines* - WAG). São três grandes Guias: i) o Guia de Acessibilidade para Ferramentas de Autoria (*Authoring Tool Accessibility Guidelines* - ATAG); ii) Guia de Acessibilidade para Conteúdo da Web (*Web Content Accessibility Guidelines* - WCAG) e o iii) o Guia de Acessibilidade para as ferramentas de usuário (*User Agent Accessibility Guidelines* - UAAG).

O ATAG é um guia para a produção de ferramentas de autoria que auxiliem os executores/construtores de conteúdo a criar conteúdos compatíveis com as características de acessibilidade. Estas ferramentas incluem editores de HTML, de texto, de imagens, de *scripts* e outros elementos que irão compor as páginas e os serviços a serem publicados.

Apesar da aceitação do mercado das diretivas propostas pelo W3C, inclusive porque grandes empresas de software fazem parte da organização e de seus comitês, ainda são poucas as ferramentas de autoria que recebem a chancela W3C-WAI[3]. O próprio W3C apoia o desenvolvimento de uma ferramenta em software livre, disponível para plataformas Unix e Windows, chamada Amaya[4], para tentar suprir a demanda por ferramentas compatíveis. Mesmo as ferramentas muito difundidas necessitam de complementos (ou *plug-ins*) para conseguir produzir conteúdo acessível (AQUINO; CRISTINA, 2009).

Com a ampliação de serviços de computação nas nuvens, novas formas de produção de conteúdo surgem naturalmente e com frequência. Os Sistemas Gestores de Conteúdo (*Content Management Systems* - CMS), como o Joomla[5], Drupal[6] e Plone[7], são bons exemplo. Os CMS permitem que qualquer usuário com pouca ou nenhuma experiência no desenvolvimento para Web construam sites complexos, com elementos textuais e multimídia, páginas comerciais, informativas e de fornecimento de serviços, que sem os devidos cuidados são potencialmente portadoras de não-conformidades quanto à acessibilidade (LÓPEZ *et al.*, 2011).

O UAAG qualifica as ferramentas de apresentação de conteúdo para os usuários, estabelecendo as características que devem ser apresentadas aos usuários quando o conteúdo multimídia de uma página da Web é lida. Três desafios são potencialmente problemáticos para o atendimento do UAAG: i) a integração de mídias distintas (arquivos com formato PDF ou DOC, por exemplo, e plataformas multimídia como o Adobe Flash[8] ou Microsoft Silverlight[9]); ii) a integração com ferramentas assistivas, como leitores de tela e conjuntos de caracteres de línguas distintas (WATANABE; UMEGAKI, 2007); e iii) a multiplicidade de plataformas de hardware (dentre elas, dispositivos móveis) (TREWIN, 2006).

O ATAG e UAAG dependem diretamente da Indústria de produção de software e consequentemente de demandas de mercado. Muito embora os padrões internacionalmente

[3] http://www.w3.org/WAI/AU/2002/tools.html
[4] http://www.w3.org/Amaya/
[5] http://www.joomla.org
[6] http://www.drupal.org
[7] http://www.plone.org
[8] http://www.adobe.com/br/products/flash/
[9] http://www.microsoft.com/brasil/silverlight/

aceitos apontem para a incorporação de características de acessibilidade, as disputas de mercado muitas vezes levam para destinos não esperados.

Por fim, o WCAG, cuja versão 2.0 foi lançada em 2008, apresenta a compatibilidade de conteúdo e forma da apresentação deste conteúdo. Essencialmente é dotado de: i) Princípios (perceptível, operável, compreensível e robusto); ii) Recomendações (disponibilizam a estrutura e os objetivos de âmbito geral para ajudar os autores a compreender os critérios de sucesso e a implementar melhor as técnicas); iii) Critérios de Sucesso (critérios testáveis de forma a permitir que as recomendações sejam utilizadas, e para que se satisfaçam um dos três níveis de conformidade, a saber A, AA e AAA, onde o primeiro é mais baixo e o último mais elevado) e iv) Técnicas (conjunto de boas práticas de formatação, sintaxe de funções e uso de recursos).

Para cada uma das recomendações e critérios de sucesso existentes foram documentadas técnicas específicas. As técnicas têm caráter informativo e enquadram-se em duas categorias: as que são do tipo suficiente, ou seja, suficientes para atender aos critérios de sucesso e as do tipo aconselhada. Estas últimas vão além do que é requerido em cada um dos critérios de sucesso e permitem aos autores um melhor cumprimento das recomendações. Por exemplo, possuir um caixa exclusivo para idosos é suficiente, mas ter cadeiras na fila deste caixa é aconselhado. Algumas técnicas de tipo aconselhadas permitem gerar uma satisfação superior à apontada pelos critérios de sucesso testáveis (CHISHOLM, W. *et al.*, 2001).

Embora o WCAG 2.0 seja amplamente aceito, existem discordâncias como o WCAG Samurai[10], que é um conjunto de complementos ao WCAG 1.0, criado por antigos membros do WAI-W3C, que diverge do WCAG 2.0 em alguns pontos técnicos, mas especialmente numa questão essencial: para os criadores do WCAG Samurai, não devem haver recomendações, pois recomendações não são deveres, mas sugestões. Para o grupo dissidente, acessibilidade deve ser uma obrigação e não uma alternativa. Este ponto, de vista embora não seja amplamente aceito, parece andar paralelamente com a postura legal de alguns governos ao determinar a obrigatoriedade de caracteristicas de acessibilidade nos conteúdos, como forma de garantia de direitos essenciais dos cidadãos (RIBERA *et al.*, 2009).

Os princípios técnicos descritos no WCAG, ao serem aplicados ao conteúdo, são tecnicamente rastreáveis, testáveis e verificáveis, seja por humanos seja por software capaz de interpretar o conteúdo. Nesta linha de raciocínio, existem dezenas de ferramentas de validação disponíveis, algumas proprietárias, mas a maioria gratuita, provida muitas vezes por governos.

[10] http://www.wcagsamurai.org/

O próprio W3C fornece não só ferramenta de validação como permite que se agregue ao conteúdo *link* para validação. Possui também uma lista com mais de uma centena de ferramentas de validação[11].

Contudo estas ferramentas tratam da validação, mas não incentivam a execução do ciclo iterativo incremental sugerido pelo WAI. Assim, o Torossou não pretende apenas integrar a lista de ferramentas disponíveis, pois ao integrar-se no ciclo de desenvolvimento passa a atuar de forma distinta das ferramentas já reconhecidas.

Em resumo, o WAI trata de padronizar características, bem como elementos técnicos e humanos necessários para que se possa mensurar e testar a acessibilidade, e faz isso através de *Guidelines* e especificações técnicas, conforme ilustrado na Figura 3.

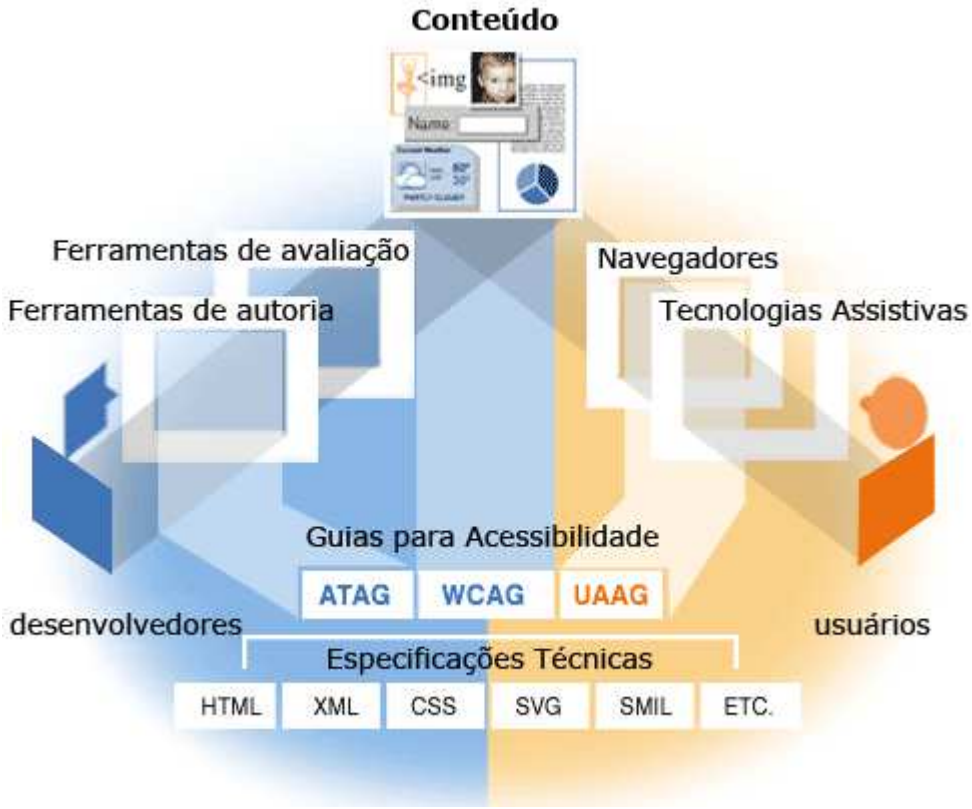


Figura 3 - Visão geral do WAI-W3C. Traduzido de: www.w3c.org.

Dentro deste contexto o Tohossosu, enquadra-se dentre as ferramentas de validação, fortemente vinculada ao ciclo da criação do conteúdo.

[11] http://www.w3.org/WAI/ER/tools/complete

2.3 Web Semântica e Ontologias

Outra iniciativa do W3C é permitir que a Web deixe de ser uma rede de documentos e torne-se uma rede de dados/informações, passando a tratar significado. Esta iniciativa recebeu o nome de Web Semântica (BERNERS-LEE; HENDLER, 2001; SHADBOLT *et al.*, 2006). Para os computadores, normalmente, palavras são meros conjuntos de caracteres, especialmente se não fazem parte de códigos ou textos construídos para serem interpretados por máquinas. Por outro lado, para os seres humanos, palavras iguais podem ter significados distintos em contextos distintos, e palavras diferentes podem ter o mesmo significado. Portanto, não é trivial para máquinas entender o significado das coisas. Desta forma, o conteúdo da Internet, criado por pessoas, mas manipulado por computadores, não é compreendido de fato pelas máquinas (FENSEL, DIETER *et al.*, 2011). Ao acrescer significado à Internet, a Web Semântica busca fazer com que o conteúdo, hoje acessível às pessoas, estejam também acessíveis aos computadores, tendo como impacto mais imediato o incremento na qualidade nos resultados dos mecanismos de buscas (HOGAN *et al.*, 2011).

Diante da evolução dos computadores, os mesmos têm sido usados como ferramentas de manipulação de dados, mas sempre com a interferência do homem quando se trata da manipulação de dados subjetivos. Desta forma, tem sido um desafio para os cientistas da computação permitir que os sistemas computacionais sejam capazes de interpretar condições subjetivas (VALLVERDÚ, 2010).

As ontologias aparecem como tecnologia concreta e promissora para vencer este desafio. De acordo com Uschold e King (1995), uma ontologia é uma especificação de uma conceituação, ou seja, uma descrição de conceitos e relações que existem em um domínio de interesse. Basicamente, uma ontologia consiste de conceitos e relações, e suas definições, propriedades e restrições, descritas na forma de axiomas, sendo compreensíveis a humanos e máquinas (USCHOLD; KING, 1995).

Desta forma, as ontologias oferecem uma forma de tratar a representação de recursos de informação. O modelo de domínio descrito por uma ontologia pode ser usado como uma estrutura unificadora para prover semântica e uma representação comum à informação (DAVIES *et al.*, 2003). Ontologias têm se tornado populares, em grande parte, pelo fato de terem como objetivo promover um entendimento comum e compartilhado sobre um domínio, que pode ser comunicado entre pessoas e sistemas de aplicação (HENDLER, 2001).

Uma ontologia envolve um vocabulário de representação, que captura os conceitos e relações em algum domínio, e um conjunto de axiomas, que restringem a sua interpretação (GUARINO, 1998).

O potencial do uso de ontologias para lidar com o problema da semântica de recursos de informação, sobretudo quando há grandes volumes de informação, tem sido largamente explorado pelas áreas de pesquisa da Web Semântica (SHADBOLT *et al.*, 2006) e da Gestão de Conhecimento (STAAB *et al.*, 2001), onde esse problema é claramente crucial. Por isso ontologias tornaram-se a essência da Web Semântica definida pelo W3C.

2.3.1 Formalização das ontologias

Uma vez que as ontologias devem não só representar conceitos mas, acima de tudo, serem compreensíveis ou passíveis de processamento por máquinas, é necessário que estas representações de conhecimento possuam um conjunto de características específicas, que culminam numa especificação formal. Dentre as especificações que se tornaram mais aceitas estão àquelas propostas pelo W3C, ou seja, o *Resource Description Framework* (RDF[12]) e a *Web Ontology Language* (OWL[13]). Um dos motivos desta aceitação é a universalidade da estrutura destes dois padrões.

Ambas possuem uma estrutura baseada em classes, que representam conjuntos de indivíduos ou, mais formalmente, representações concretas de conceitos. Cada classe tem propriedades (características) com valores individuais ou múltiplos e com restrições próprias. As classes relacionam-se hierarquicamente entre si (taxonomia). Os relacionamentos entre as classes são determinados por regras explicitas como, por exemplo, restrições de tipo, integridade, quantidade, dependência, herança, ordem, cardinalidade, dentre outras.

Um indivíduo específico pertencente a uma classe, com suas propriedades únicas, constitui uma instância. O indivíduo é membro daquela classe e herda desta última todas as suas características gerais. Contudo, dois indivíduos ou instâncias da classe podem de fato representar um mesmo conceito real. Por exemplo, poder-se-ia ter a classe das “raízes comestíveis”; da qual “beterraba” seria um indivíduo, assim como “batata”, “mandioca” e “macaxeira”, mas estas duas últimas, apesar de serem instâncias distintas, representam o

[12] http://www.w3.org/RDF/

[13] http://www.w3.org/OWL/

mesmo elemento concreto. Esta relação de igualdade deve ser então explicitada através de uma relação entre estas duas instâncias (HORRIDGE, 2009).

2.3.1.1 RDF

Em 2004, o W3C lançou o padrão RDF para facilitar a interoperação entre aplicações na Web, através da representação de metadados acerca dos recursos oferecidos e requeridos pelas aplicações. Uma vez que não há restrições sobre a natureza destes metadados, o W3C percebeu que o RDF poderia ser empregado para representar informações sobre qualquer recurso que pudesse ser identificado na Web por meio de marcadores, chamados de *Uniform Resource Identifiers* (URI[14]).

O RDF utiliza URIs para identificar indivíduos, tipos de coisas, propriedades e os valores dessas propriedades. Através de URIs, pode-se estabelecer um relacionamento entre quaisquer dois elementos do RDF. Isto é feito por meio de uma sentença com sintaxe *sujeito* + *predicado* + *objeto* (ex. "beterraba" + "é uma" + "raiz comestível") ou por uma sentença descrevendo *objeto* + *propriedade* + *valor*. Valores podem ser outras URIs ou valores literais.

A estrutura hierárquica, bem como a generalização e especialização, entre classes no RDF é dada pelo *RDF Schema* (RDFS). O RDFS permite uma interpretação do RDF de forma muito similar aos conceitos de Orientação a Objeto (OO), tão conhecidos dos produtores de software, e possui os seguintes elementos constituintes: i) *rdfs:Resource*, onde todas as "coisas" descritas são consideradas recursos ou objetos; ii) *rdfs:Property*, que representam propriedades ou características; iii) *rdfs:Class*, onde classe é um conceito similar à OO; iv) *rdfs:type*, que indica que um dado recurso é membro de uma classe; v) *rdfs:subClassOf*, onde sub-classe também é um conceito similar à OO, incluindo o conceito de herança; e vi) *rdfs:subPropertyOf*, que indica que uma propriedade é especialização de outra.

Considera-se que a formalização dada pelo RDFS é que torna o RDF apropriado para a construção de ontologias, pois é nele que se delineiam as características de semântica.

Para a representação escrita e armazenamento do RDF, o W3C elegeu um outro padrão bastante maduro, a *Extensible Markup Language*, ou XML. Este padrão, dentre outras

[14] Padrão IETF especificado pela RFC3986. http://www.ietf.org/rfc/rfc3986.txt

vantagens, pode ser lido e editado por máquinas e é compreensível por pessoas. Esta última característica é especialmente importante quando se lida com a representação de conhecimento. A Figura 4 apresenta uma ontologia, na forma de grafo, em RDF.

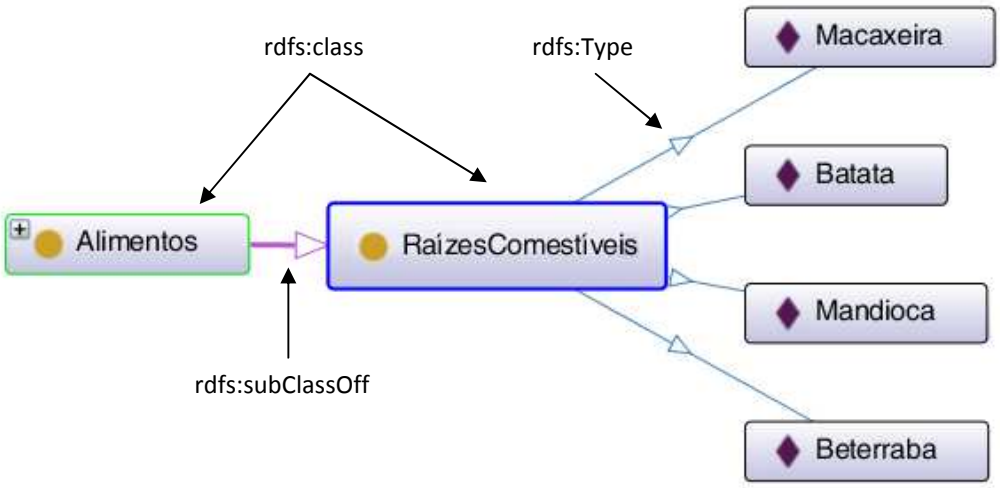


Figura 4 - Grafo representativo da uma ontologia simples escrita em RDF.

Na Figura 4 tem-se a representação de uma ontologia que descreve alimentos, onde a classe “RaízesComestíveis” é uma sub-classe da classe “Alimentos”. Beterraba, mandioca, batata e macaxeira são instâncias (tipos) da classe “RaízesComestíveis”. Esta mesma ontologia está descrita através de XML no Quadro 01.

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
   <!ENTITY AlimentosVegetais "http://www.semanticWeb.org/ontologies/2012/6/AlimentosVegetais.rdf#" > ]>
<rdf:RDF xmlns="http://www.semanticWeb.org/ontologies/2012/6/AlimentosVegetais.rdf#"
    xml:base="http://www.semanticWeb.org/ontologies/2012/6/AlimentosVegetais.rdf"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:AlimentosVegetais="http://www.semanticWeb.org/ontologies/2012/6/AlimentosVegetais.rdf#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
   <rdfs:Class rdf:about="Alimentos"/>
   <rdfs:Class rdf:about="RaízesComestíveis">
      <rdfs:subClassOf rdf:resource="Alimentos"/>
   </rdfs:Class>
   <rdf:Description ID="Batata">
      <rdfs:type rdf:resource="RaízesComestíveis"/>
   </rdf:Description>
   <rdf:Description ID="Beterraba">
      <rdfs:type rdf:resource="RaízesComestíveis"/>
   </rdf:Description>
   <rdf:Description ID="Macacheira">
      <rdfs:type rdf:resource="RaízesComestíveis"/>
      <rdf:sameAs rdf:resource="Mandioca"/>
   <rdf:Description>
   <rdf:Description ID="Mandioca">
      <rdf:type rdf:resource="RaízesComestíveis"/>
      <rdf:sameAs rdf:resource="Macacheira"/>
   <rdf:Description>
</rdf:RDF>
```

Quadro 01 – Representação RDF/XML de Ontologia

2.3.1.2 OWL

Assim como o RDF, a OWL é uma linguagem usada para representar ontologias, escrita em XML, que pode ser visualizada em gráficos. Contrariamente ao RDF, que precisou ser estendido através do RDFS, a OWL possui nativamente todas as funcionalidades do RDF/RDFS e inclui outras características ausentes na primeira, como a possibilidade de construção de axiomas, ou seja, sentenças lógicas que representam verdades sem exceções para as instâncias e classes.

O Projeto da OWL foi concluído em 2004, baseada em outro padrão, o DAML+OIL, que por sua vez foi construído da intersecção de outras duas linguagens de ontologia, a DAML (*DARPA Agent Markup Language*[15]) e a OIL (*Ontology Inference Layer* (FENSEL, D. *et al.*, 2001; ANTONIOU; HARMELEN, FRANK VAN, 2009)).

O W3C revisou a especificação da OWL em 2009, tendo como produto final a especificação da OWL2[16], que trás algumas melhorias, como restrições de cardinalidade e faixas de valores para propriedades, sem, contudo, perder a interoperação com a versão anterior e principalmente sem a perda do vínculo com o RDF, como se pode observar na Figura 5.

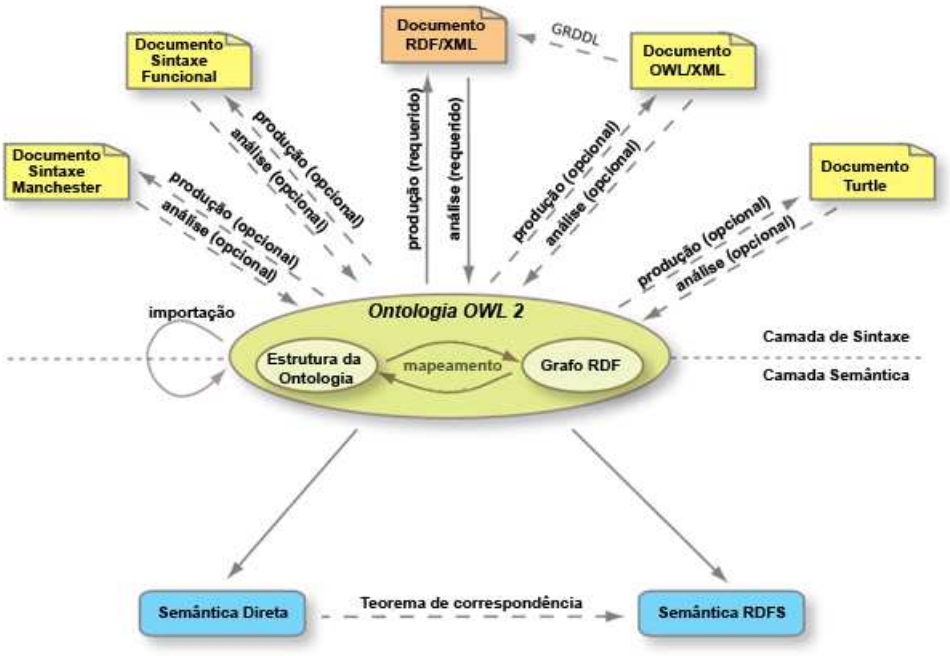


Figura 5 - Estrutura da OWL2. Fonte: http://www.w3.org/TR/owl2-overview/

[15] DARPA é a *Defense Advanced Research Projects Agency*, reconhecida como uma das responsáveis pela criação da infraestrutura da Internet (LEINER *et al.*, 1997).

[16] http://www.w3.org/TR/owl2-overview/

2.3.2 Aplicações em Engenharia de Software e Acessibilidade

Durante um projeto de software, em especial para aplicações para Web, muitas informações são produzidas e requeridas e, em muitas situações, é essencial estabelecer conexões entre recursos para se obter o conjunto necessário de dados para apoiar a realização de uma atividade ou a criação de um artefato. Neste contexto, as ontologias são potenciais ferramentas para apoio no desenvolvimento de aplicações compatíveis com bases de conhecimento claramente definidas (WONGTHONGTHAM, P. *et al.*, 2006).

Trabalhos demonstram que a aplicação de ontologias nas atividades inerentes à Engenharia de Software, seja no processo de planejamento e construção de aplicações, seja nas etapas de definição de requisitos, geram resultados positivos e facilitam a execução das atividades de codificação e a correlação dos requisitos nos produtos (ZHANG; LEE, B., 2010). As dificuldades maiores surgem quando os serviços geram conteúdos dinâmicos, presentes na grande maioria dos serviços atuais. Nestes casos, estão sendo propostas metodologias específicas para automatizar a criação dos metadados pertinentes à acessibilidade (PLESSERS *et al.*, 2005).

O uso de ontologias para auxiliar no mapeamento e na documentação de requisitos para registro de características dos produtos que atendam demandas de acessibilidade é um tema de pesquisa que tem atraído o interesse de pesquisadores , pois há o entendimento que a disciplina de obtenção de requisitos em toda sua complexidade tem dificuldade de atender aos requisitos de acessibilidade (MASUWA-MORGAN, K.R; BURRELL, 2004; MASUWA-MORGAN, KRISTINA RUNGANO, 2008).

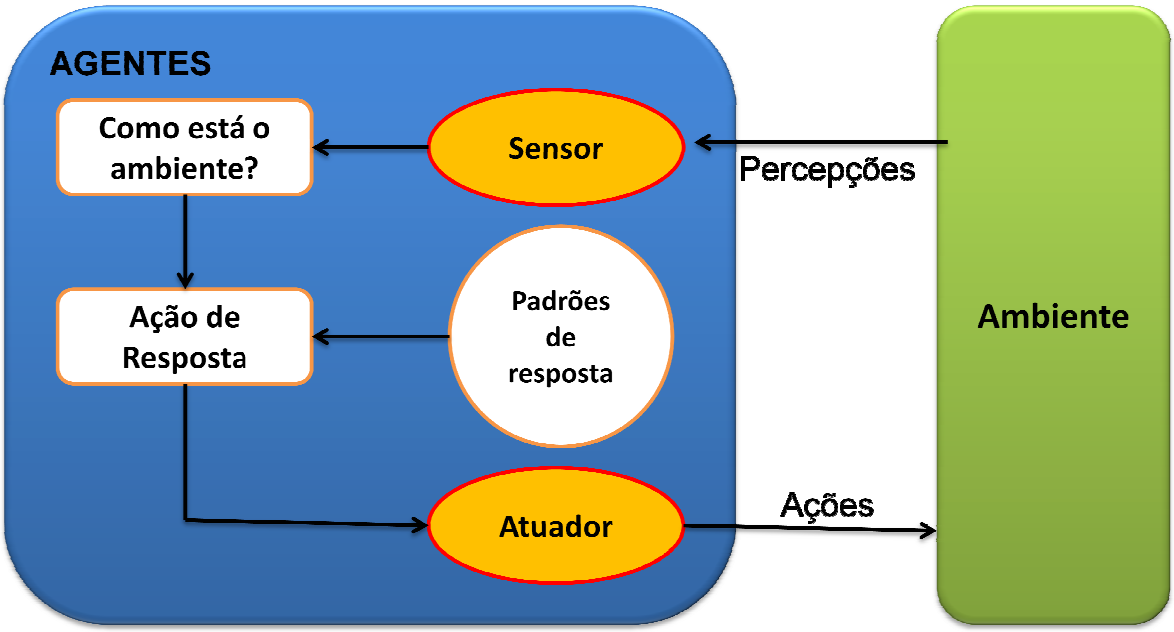
Por outro lado, o uso automatizado de ontologias tem tomado muitos caminhos e um dos mais promissores é a aplicação conjunta com agentes inteligentes. Prover, aos agentes de software, bases de conhecimento e argumentos para tomadas de decisão através de ontologias possibilita a expansão constante do conhecimento e uma forte interação com pessoas para revisão e aprimoramento do mesmo. De Serviços Web (GARCÍA-SÁNCHEZ *et al.*, 2009) a sistemas especialistas em medicina (WANG *et al.*, 2010), passando por ferramentas de apoio ao aprendizado (FONTES *et al.*, 2011), o uso de sistemas multiagente e ontologias são mais que uma tendência, mas uma alternativa concreta.

O uso de ontologias em ambientes multiagente é uma realidade cada vez mais presente (HADZIC *et al.*, 2009). As possibilidades desta interação agente-ontologia criam cenários de resultados complexos, como, por exemplo, ambientes colaborativos de apoio ao ensino

(empregados no ensino presencial e especialmente no ensino a distância), onde os alunos têm seu aprendizado orientado por agentes, que têm sua estrutura de conhecimento baseada em ontologias (PONTES, A. Á., 2010).

2.4 Agentes Inteligentes

Definir o que seja um agente inteligente é complexo, porque complexo é definir o que é inteligência (CANDEIAS *et al.*, 2008). Em virtude disto, as definições de agentes inteligentes são norteadas principalmente por suas capacidades e suas possibilidades de tomadas de ação e decisão. Uma definição clássica de agentes é aquela dada por Peter Norvig e Stuart Russel (2010), em que um agente é um componente capaz de perceber seu ambiente, através de sensores, e alterar este ambiente, por meio de atuadores. Com este pensamento, seres humanos e os outros seres vivos são todos agentes. Esta similaridade entre os seres vivos e os agentes, por si só, já sugere uma infinidade de aplicações para a tecnologia (RUSSELL; NORVIG, 2010).



Related and derived concepts include Intelligent agents (in particular exhibiting some aspect of Artificial Intelligence, such as learning and reasoning), autonomous agents (capable of modifying the way in which they achieve their objectives), distributed agents (being executed on physically distinct computers), multi-agent systems (distributed agents that do not have the capabilities to achieve an objective alone and thus must communicate), and mobile agents (agents that can relocate their execution onto different processors).

Além de perceber e atuar, outra característica aprimora o conceito de agente: agir com racionalidade. Neste caso tem-se um agente dito inteligente. Muito embora existam discussões que definam que agir com racionalidade não obrigatoriamente signifique agir corretamente. É consenso que um agente capaz de agir segundo uma dada lógica é uma ferramenta poderosa, pois pode assumir o papel de agir, sozinho ou coletivamente, com regularidade, confiabilidade e de forma autônoma (RUSSELL; NORVIG, 2010).

Neste sentido, para agir com lógica, os agentes devem possuir conhecimento, estático ou que se aprimore com o tempo, e objetivos. Com estes requisitos, as decisões racionais são mais bem conduzidas, por mais simples que seja o agente ou o ambiente onde o mesmo encontra-se (ARTERO, 2009).

Esta definição é importante porque permite perceber que estas ações autônomas que o agente pode e deve tomar são decorrentes da base de conhecimento que o mesmo tem disponível. Neste ponto as ontologias surgem como alternativa para estruturar a base de conhecimento dos agentes inteligentes (LI *et al.*, 2005).

2.4.1 Características e tipologia dos agentes

Se definir o que seja agente inteligente não é uma tarefa fácil, que por vezes pode gerar divergência, caracterizá-los e tipificá-los é igualmente difícil e, mais uma vez, a literatura não trás um consenso, mas diversos conceitos que se complementam. Uma dificuldade mesmo dentre os desenvolvedores é distinguir os agentes inteligentes de outras entidades de software (FRANKLIN; GRAESSER, 1997)

Uma classificação muito aceita é aquela que se refere ao mecanismo com que o agente responde a estímulos ambientais[17]. Se o agente reage ao ambiente baseado apenas no estado

[17] Considera-se aqui “ambiental” como o ambiente externo ao agente, independentemente de sua natureza.

atual, ou seja, no estado interno do agente e no resultado das percepções instantâneas do ambiente, sem considerar as ações passadas, ele é chamado de Agente Reativo Simples. Se, por outro lado, o agente obtém o estado do ambiente e armazena estes dados para interpretar, ou criar um modelo do ambiente, para somente então reagir, é chamado Agente Reativo Baseado em Modelos (RUSSELL; NORVIG, 2010).

Outros agentes, além de observar o ambiente, respondem também considerando determinados objetivos ou planejamentos. Estes objetivos podem ser previamente determinados ou ajustados com o tempo. O Planejamento para atingi-los pode ser alterado conforme muda o ambiente, mas as ações do agente sempre serão guiadas no sentido de aproximar-se do objetivo traçado. Estes são chamados de Agentes Baseados em Objetivos. Por fim, se o agente possui mecanismos de buscar "ganhos" ou vantagens competitivas, para atingir a certos objetivos, ou mesmo para poder tomar a decisão de realizar uma operação que gere uma recompensa antes de atingir ao objetivo, o agente é dito Baseado em Utilidades (RUSSELL; NORVIG, 2010).

Independentemente desta tipificação, algumas características são úteis no momento de identificar um agente. Obviamente nem todas as características podem ou devem estar presentes simultaneamente. Um agente deve gozar de Autonomia, ou seja, poder realizar ações de forma independente, como resultado de suas decisões ou reações. Embora possa haver interferência de outros entes externos, inclusive pessoas, espera-se que o agente possa determinar sozinho suas formas de ação. Qualquer que seja o ambiente (estático, dinâmico, determinístico ou aleatório) e a forma de captá-lo, o agente somente será útil se souber e puder reagir aos estímulos do ambiente de forma apropriada. Além disso, espera-se que em alguns casos os agentes possam ser Pró-ativos, e apresentando respostas distintas a estímulos e mudanças distintas (FRANKLIN; GRAESSER, 1997).

Se o agente é capaz de mensurar o resultado de suas ações, qualificando-as e elaborando respostas novas aos estímulos, esta capacidade é chamada de Adaptação. O agente pode ainda ter a capacidade de perpetuar suas ações ao longo do tempo, ainda que não de forma contínua. Isso implica em poder armazenar seu estado, suas informações do meio e seus planos e objetivos, para que mediante uma pausa, parada, ou outro evento esperado ou não, ele possa continuar suas ações. Decidir quando vai parar ou continuar definem também a característica de Continuidade temporal (FRANKLIN; GRAESSER, 1997).

Se o ambiente permitir, o agente de software pode ter Mobilidade. Neste caso, mobilidade refere-se a poder de fato transitar entre ambientes diferentes (possivelmente até mesmo hardware diferente) e agir seguindo os mesmos critérios (ou adaptando-se à nova

realidade). Espera-se que os agentes sejam Confiáveis, ou seja, atuem de forma compreensível e determinada, ou, segundo alguns autores, possam agir com a verdade e, se são inteligentes, que possam otimizar os recursos à sua disposição, maximizando os resultados e atingindo os objetivos com o máximo de Racionalidade (WOOLDRIDGE; JENNINGS, 1995).

Por fim, duas características são determinantes na qualidade das ações de ambientes onde existem mais de um agente: Interação e Cooperação.

Nestes casos é essencial que o agente possua habilidades sociais ou capacidade de Interação (WOOLDRIDGE; JENNINGS, 1995). A interação com o ambiente já é algo inerente, mas aqui se busca a interação com outros agentes computacionais ou mesmo humanos. Esta interação obviamente explicita a necessidade de comunicação, seja direta (com linguagem própria ou intercâmbio de dados e metadados, por exemplo) seja através de artifícios, como compartilhamento de artefatos ou de um quadro de mensagens (STEELS, 2003).

De forma complementar à interação é importante a capacidade de Cooperação dos agentes. Ao cooperar com outros entes, os agentes podem trabalhar conjuntamente em prol de atingir objetivos comuns, solucionar problemas dividindo-o em partes (para atingir a uma solução plena), ou mesmo sacrificando suas atividades para um melhor desempenho da coletividade (ARTERO, 2009).

2.4.2 Sistemas Multiagente

Quando o ambiente possui não um, mas vários agentes, que se relacionam entre si, este constitui um Sistema Multiagente (SMA). Os SMAs encontram-se em uma subárea da Inteligência Artificial Distribuída (IAD), que estuda o comportamento de um grupo organizado de agentes autônomos, que se comunicam, interagem e trabalham coletiva e colaborativamente na resolução de problemas que estão além das capacidades individuais (HÜBNER, 2003).

Esta abordagem colaborativa tem potencial para prover soluções em menor tempo e com menos recursos. Ao trabalhar com uma comunidade de agentes, é possível introduzir indivíduos com novas capacidades ou retirar indivíduos ociosos ou dispendiosos, promovendo o reuso de código e demais artefatos de programação. Trabalha-se ainda com o paralelismo de

ações e processamento, gerando resultados em um tempo menor do que se a atividade fosse executada de forma serial (FERBER, 1999).

Dividir problemas complexos, tratando-os como um grupo de pequenos problemas, é uma estratégia comum ao ser humano e a outros seres vivos. Nos SMAs objetiva-se que um ou um grupo de agentes trate de cada um dos pequenos problemas, aumentando assim as chances de solução do problema maior. Assim, cada agente especializa-se no domínio do pequeno problema que está tratando, enquanto o coletivo consegue solucionar com agilidade e confiabilidade o problema global (ARTERO, 2009).

Além disso, com a convivência dos agentes no mesmo meio, e com as alterações promovidas por cada um deles no ambiente, percebe-se que as ações individuais alteram o coletivo, de modo que deve haver um conjunto de regras ou restrições de convivência entre os agentes, para que o trabalho seja de fato colaborativo e não concorrente (REZENDE, 2003), por exemplo definir que uma atividade que já está sendo tratada por um agente, não seja tratada por nenhum outro. O estabelecimento de hierarquias ou classes (com designação expressa de funções para cada agente ou grupos de agentes) (LESSER, 1999), a criação de uma estrutura baseada na reputação de cada agente (critérios de sucesso, racionalidade e vantagens na solução de um problema ou parte dele) (SABATER; SIERRA, 2002) ou mesmo relação de contratação (onde um agente publica que pode prover um serviço e outro "contrata" o mesmo para executá-lo) (SYCARA, 1998) são requisitos necessários para que de fato possa se obter proveito de um SMA.

A comunidade de agentes no SMA deve possuir formas específicas de comunicar-se, gerando produtos úteis para o coletivo, e garantindo a interoperabilidade entre os agentes e entre agentes de sistemas distintos. Algumas linguagens, como a KQML (*Knowledge Query and Manipulation Language*) (WU, X.; SUN, 2010) e a ACL (*Agent Communications Language*) (AHMED *et al.*, 2011), e metodologias têm sido propostas ao longo do tempo (SYCARA, 1998), inclusive algumas baseadas em ontologias, para facilitar esta comunicação (LACLAVÍK *et al.*, 2006).

Para modelar os agentes individuais e o coletivo, desenvolveram-se metodologias específicas, cada uma com vantagens e desvantagens, dentre as quais se pode destacar a MAS-CommonKADS+, que possibilita diversos expressar detalhes da implementação, inclusive descrever os mecanismos internos dos agentes (MORAIS II, 2010).

3 Tohossou

A WAI/W3C prevê um ciclo de produção de conteúdo acessível que passa pela validação do conteúdo. As ferramentas disponíveis, que são diversas, trabalham hoje sobre o produto acabado e têm dificuldades em lidar com conteúdo dinâmico, o que gera dificuldades, pois, com o produto finalizado, se uma ou mais não-conformidades são detectadas, força-se ao retrabalho e o incremento dos custos do projeto.

Sob outra ótica, em tempos de *Cloud Computing*, torna-se cada vez mais comum que as equipes distribuídas geograficamente trabalhem de forma colaborativa em projetos de aplicações para Web. Na maioria das vezes, estas equipes são conectadas por ferramentas de gestão de projetos, redes sociais e, principalmente, controle de versões de código (CVS – *Control Version System*, cuja ferramenta mais conhecida é o SubVersion, ou SVN[18]) e as ferramentas tradicionais não conseguem realizar este tipo de integração com as ferramentas de garantia de acessibilidade.

Estes, dentre outros fatores, como a falta de conhecimento técnico em relação aos padrões de acessibilidade, impactam negativamente na criação de conteúdo acessível. O Tohossou apresenta-se como uma ferramenta com potencial para alterar este cenário, permitindo que o ciclo de produção abranja todas as etapas de criação dos conteúdos acessíveis.

3.1 Caracterização

Tohossou (lê-se tu-rru-ssú, pronúncia influenciada pela colonização francesa) é uma deidade africana que, segundo a tradição de alguns povos residentes hoje em Gana e no Togo, protege os nascidos com deficiências. Conta a lenda que todas as crianças nascidas com deformidades deveriam ser sacrificadas, mas Tohossou as abrigava e lhes dava uma vida normal, até que uma grande guerra afligiu a região e a vitória somente chegou quando Tohossou trouxe todos os seus protegidos para batalhar. Desde então, para estes povos, os

[18] http://subversion.apache.org/

nascidos com características diferentes, longe de serem um problema, são sinal de bons augúrios.

Tohossou foi o nome escolhido para o SMA criado para auxiliar no suporte à acessibilidade em aplicações para Web, interagindo com a equipe de projeto e desenvolvimento destas aplicações.

Para cumprir o seu papel, o Tohossou possui uma série de características que lhe permitem ir além das ferramentas em uso na atualidade. Acima de tudo, é um SMA, onde cada agente, especializado em determinada atividade, trabalha colaborativamente com os demais, comunicando-se com frequência com outros agentes e com os usuários do sistema.

É uma aplicação para Web, com armazenamento de dados e gestão dos agentes executada no servidor do serviço, dentro da visão de *cloud computing*, não necessitando de infraestrutura local, exceto o acesso à Internet, de forma que seu uso não onera o projeto nem a equipe de desenvolvimento no sentido de manter mais um software ou serviço funcionando e seu respectivo hardware, recursos, insumos (energia elétrica, por exemplo) e atividades secundárias, dentre elas *backup*.

O Tohossou foi desenvolvido usando ferramentas de software livre. Foram utilizados o framework Yii[19], a linguagem PHP[20] e alguns de seus pacotes suplementares e o banco de dados relacional MySQL[21], com todas as vantagens de custos reduzidos, apoio da comunidade de desenvolvedores, portabilidade, além de facilitar alterações, criação de *plug-ins* e outras funcionalidades por terceiros, sendo possível integrá-la com as demais ferramentas de produtividade da equipe de desenvolvimento, especialmente pelo uso do *Web service* integrado à ferramenta. A interface, apesar de estar dentro dos conceitos das *Rich Internet Applications* (RIA) (FRATERNALI *et al.*, 2010), pode ser considerada limpa (*clean*), projetada para poder incorporar novos elementos visuais (logos, textos e representações gráficas dos agentes), ser personalizada se assim desejado, e manter, mesmo assim, as características de acessibilidade.

Além disso, o Tohossou foi desenvolvido para comunicar-se ao serviço SVN usado pela equipe, sendo capaz de detectar cada novo artefato inserido no mesmo, realizar as análises de acessibilidade e gerar resultados que serão encaminhados aos partícipes do projeto. Desta forma, a equipe não precisa interromper suas ações para validar código, ou aprender novas técnicas de acessibilidade (ou outras características que se deseje validar),

[19] http://www.yiiframework.com/

[20] http://www.php.net/

[21] http://www.mysql.com/

pois o Tohossou aponta a não conformidade e indica como corrigi-la, através de e-mail enviado à equipe, de forma autônoma.

Os agentes comunicam-se com os usuários, pela interface Web, por e-mail ou pelo *Web service*, mas também comunicam-se intensamente entre si. Para tanto foi empregada a técnica do *blackboard* (quadro-negro) (LI, S.; HU, 2009) em que todos os agentes têm acesso a uma porção de memória (ou espaço de armazenamento) onde podem ler e postar mensagens seguindo uma linguagem previamente determinada. No caso do Tohossou, optou-se por um conjunto elementar de mensagens, nativo da ferramenta, sem causar problemas de interoperabilidade com outras ferramentas, dado que a comunicação com elementos externos dá-se pelo *Web service* (e uso de XML) e não através do *blackboard* que é exclusivo dos agentes do Tohossou.

Por fim, os agentes do Tohossou usam ontologias como base de conhecimento. Embora não tenha sido projetado para alterar autonomamente as suas ontologias, ele foi projetado para se adaptar a novas ontologias, sugeridas pelos usuários. Desta forma, não só múltiplos domínios de conhecimento podem ser analisados, como pode haver aprimoramentos dos domínios existentes.

Em seu desenvolvimento primário, duas ontologias foram criadas e amplamente testadas: Uma relacionada ao WCAG nível AAA e outra de detecção de não conformidades para a Dislexia. São focos completamente distintos da acessibilidade mas que convivem naturalmente na base de conhecimento no Tohossou.

3.2 Cenário de Uso

O fluxo de uso do Tohussou foi organizado de forma a maximizar a interação entre a equipe e tornar mínimos os custos do processo de integração de mais uma ferramenta às já utilizadas pela equipe.

O líder do projeto, ou um membro da equipe designado para este fim, cadastra-se no Tohossou, através de um formulário simples, mostrado na Figura 6, que tem como informação mais relevante o e-mail do usuário.

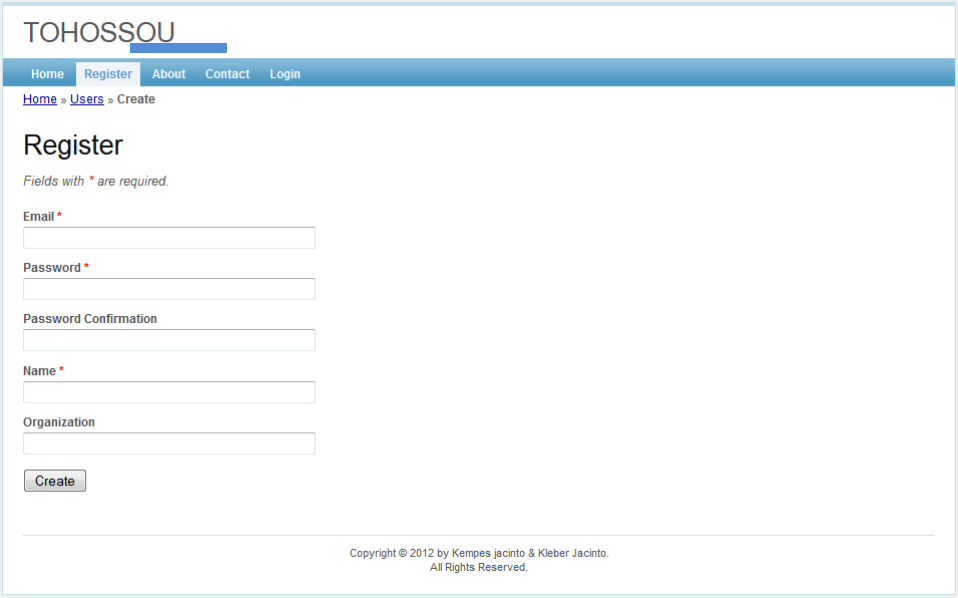


Figura 6 - Interface de cadastro de novo usuário do Tohossou

O e-mail é especialmente relevante porque o correio eletrônico é o principal canal de comunicação do Tohossou com os usuários. Já no ato do cadastro, o novo usuário recebe mensagem com um *hyperlink* para o chamado "ticket de cadastro". Este *ticket* tem validade de 10 dias e tem a finalidade de minimizar a possibilidade de cadastro não autorizado de usuários, de forma que somente serão validados cadastros cujos usuários clicarem no *link* remetido ao seu e-mail. Assim, mesmo que um terceiro realize um cadastro, somente o proprietário da conta de e-mail será capaz de validá-lo. Uma vez validado, ele pode realizar *login* na ferramenta.

O usuário deve documentar a estrutura de trabalho e interação da ferramenta. Para isso, deve (i) criar um projeto, (ii) atribuir ao projeto os repositórios que deverão ser monitorados pelos agentes e (iii) convidar os usuários que receberão as notificações, conforme ilustrado na Figura 7.

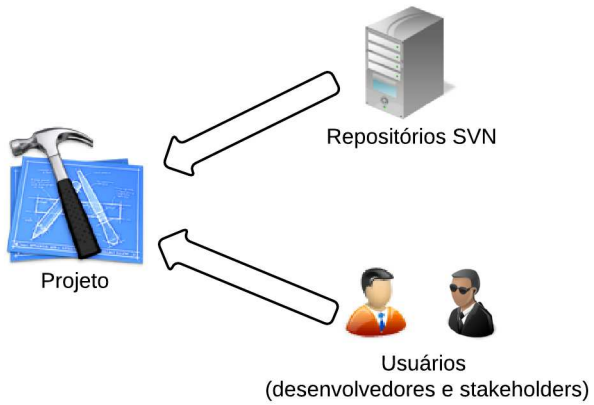


Figura 7 - Estrutura do registro de um Projeto no Tohossou.

Para o projeto, além de um nome e uma descrição, solicita-se datas de início e fim do projeto. Esta informação é necessária para que os agentes possam entender se o projeto está ou não ativo, pois a ausência de atualizações no repositório não é critério suficiente para estabelecer que o projeto não está ativo. Os agentes não usam seus recursos em projetos inativos.

O cadastro dos repositórios ocorre pela indicação da *Uniform Resource Location* (URL) e de *login* autorizado a consultar o repositório e realizar o *download* de seus artefatos. Uma vez cadastrado pelo menos um repositório, o Tohossou passa a analisar o repositório e usar os recursos de gestão de versão do SVN para buscar as versões mais recentes e não analisadas dos artefatos contidos no repositório. Os repositórios devem, obviamente, estar ao alcance dos agentes, preferencialmente na Internet. Havendo mais de um repositório em um único projeto, não se faz necessário que estejam localizados fisicamente no mesmo local ou servidor.

Os membros do projeto são convidados na própria interface pela indicação de nome e e-mail. Mais uma vez, o e-mail é a informação essencial, pois é através de mensagem que o convidado recebe o *link* para o *ticket* que libera seu cadastro. O convidado recebe por e-mail a informação de que fora chamado a participar do uso da ferramenta e o aceite deve ser feito na página indicada (*link* no corpo do e-mail). Após a confirmação, o membro passa a receber mensagens do agente de comunicação acerca do andamento do projeto, em especial quanto às não-conformidades do código publicado no repositório. Por razões de segurança, só é possível agregar-se a um projeto cadastrado via convite de um líder (ou criador) do projeto.

Com repositórios e usuários cadastrados, os agentes dos Tohossou passam a atuar e podem emitir relatórios e alertas aos usuários e aos líderes de projeto. O ciclo de trabalho do Tohossou sobre um projeto é limitado pela existência de novos artefatos no repositório e pela data de término do projeto. O fluxo de trabalho e registro dos atores envolvidos no Tohossou é apresentado na Figura 8. O detalhamento das ações dos agentes do Tohossou será explicitado nas seções seguintes.

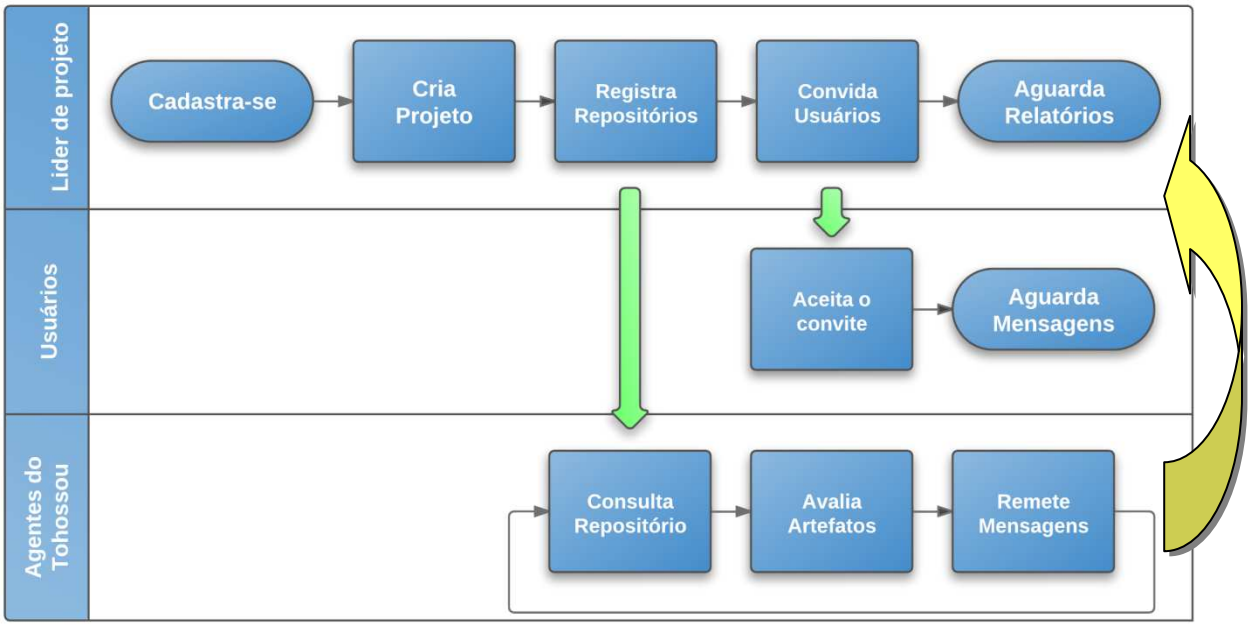


Figura 8 - Fluxo de trabalho e ações de cada ator no Tohossou

Como pode ser visto na Figura 8, os usuários (engenheiros de software, desenvolvedores, *stakeholders* e demais participantes do projeto) possuem poucas ações, o que minimiza o aprendizado, facilita a integração da ferramenta à rotina de trabalho da equipe e permite a todos concentrarem-se nos aspectos inerentes ao desenvolvimento. Eles têm como responsabilidade apenas ler as recomendações e executá-las no momento apropriado. A intenção é que o Tohossou atue como um consultor da equipe, sendo capaz de repassar informações à equipe com o mínimo possível de alterações na rotina de trabalho de todos.

O líder do projeto recebe, também por e-mail remetido pelo Agente de Comunicação, que será explicado na próxima seção, relatórios periódicos acerca da atividade de inclusão de artefatos no repositório e sobre as ações dos desenvolvedores e demais membros do projeto no sentido de corrigir as não-conformidades. Na Figura 9, há um exemplo de mensagem remetida pelo Tohossou a um líder de projeto.

From: **Tohossou Project** <tohossou@gmail.com>
Date: Sat, May 26, 2012 at 4:43 AM
Subject: Accessibility Analysis
To: **********

Accessibility Analysis
Saturday 26th of May 2012 04:43:38 AM

Accessibility Analysis

This is a automatic report about accessibility problems in Tohossou project.
Analysis was done in each repository and file registered in system, as shown bellow.

Repository	File	Problems
LES	**trunk/Arquivo_de_teste 2.html**	**BANNED Sentences** • a Tag BLINK gera efeito visual não percebido pelos leitores de tela e é confuso inclusive para leitores sem déficits de percepção visual(Sentence:<blink>) **Sentence REQUIRES ...** • alt= - é necessário prover acesso alternativo a conteúdos extritamente visuais(Sentence:<img src="www.google.com">)
LES	**trunk/Arquivo_de_testes1.html**	**REQUIRED Sentences** • **Todo documento HTML deve possuir um corpo (BODY).** **Sentence MUST HAVE ...** • **id= - (Sentence:<input valeu="teste1" />)** • **id= - (Sentence:<input valeu="teste3" />)** • **id= - (Sentence:<input valeu="teste4" />)**
LES	trunk.htaccess trunkcommunicationAgent Index.php trunkindex-test.php trunkindex.php trunkREADME.txt	This files were not analyzed

This is a automatic message created and sent by Tohossou Project, please do not answer it.

Figura 9 - Relatório emitido ao líder do projeto após avaliação no Tohossou

3.3 Comunidade de agentes do Tohossou

O Tohossou é composto de uma comunidade de agentes que detectam alterações no ambiente e executam suas ações, comunicando-se com os usuários através de e-mail da e interface Web do Tohossou e com os outros agentes através de um repositório de mensagens centralizado (*blackboard*).

O sistema possui três tipos de agentes distintos: o Agente Bibliotecário, os Agentes Analisadores[22] e o Agente de Comunicação, responsável por contatar os usuários e outros

[22] Existem mais de uma instância do agente analisador, possibilitando a análise em paralelo de vários artefatos.

sistemas. A interação entre estes agentes e os demais atores envolvidos (humanos e outros sistemas) está apresentada graficamente na Figura 10. As setas brancas representam leituras de dados; as laranjas, a comunicação entre os agentes; e a azul, a comunicação do sistema com os usuários finais.

Figura 10 - Esquema básico de ação dos agentes do Tohossou

3.3.1 Agente Bibliotecário

O Agente Bibliotecário tem a missão de acessar os repositórios de projetos e informar à comunidade de agentes do Tohossou se há material a ser validado. O agente analisa os repositórios, coletando dados fornecidos pelo SVN acerca dos arquivos encontrados, especialmente as informações relacionadas à data e à hora de atualização e ao tipo de arquivo. A data e a hora definem a linha temporal das versões dos arquivos para que o Agente Bibliotecário relate à comunidade apenas as últimas versões, pois dados de versões anteriores são irrelevantes sob a ótica da validação. Cabe relatar que, no conceito do SVN, todas as versões editadas de um mesmo arquivo são sempre mantidas para referencias futuras, para verificar as diferenças entre as versões e eventualmente realizar um *roolback*, ou seja, recuperar uma versão anterior.

O Agente Bibliotecário monitora também se houve retorno a *branchs* (linhas de desenvolvimento alternativas armazenadas no repositório) anteriores. É importante analisar também o tipo de arquivo (identificado pela extensão) para que o Bibliotecário possa estabelecer a correlação entre o arquivo e uma ou mais ontologias de validação. Contudo, para fins de avaliação de acessibilidade e outras características de conteúdo, somente a última versão é relevante. Se esta última for trabalhada e as não-conformidades forem corrigidas, todas as versões que dela descenderem estarão corretas.

Assim, o Agente Bibliotecário (i) acessa proativa e periodicamente os repositórios, em busca de alterações, acréscimos ou exclusões de arquivos, (ii) constrói uma biblioteca de artefatos e (iii) enumera dentre eles quais são passíveis de validação, quais as ontologias a serem utilizadas e quais serão ou já foram analisados.

O Agente Bibliotecário gerencia ainda o repositório local das ontologias que serão usadas pelos Agentes Analisadores. Esta responsabilidade deve-se pela possibilidade de inclusão de novas ontologias ou alteração das ontologias existentes. Para que fosse garantida a possibilidade de intercâmbio e acréscimo de novas ontologias, decidiu-se pelo uso de ontologias em formato RDF, que será detalhado na Seção 3.4. O Quadro 02 apresenta o *template* textual[23] do Agente Bibliotecário, que descreve suas principais características.

Agente Bibliotecário
Objetivo: Manter sempre o máximo possível de artefatos analisados.
Parâmetros de Entrada: Projetos cadastrados na interface Web com seus respectivos repositórios.
Parâmetros de Saída: Catalogação dos artefatos do projeto e mensagens para o Agente Analisador.
Condições de Ativação: Alterações no repositório de artefatos do projeto.
Condições de Finalização: A catalogação foi finalizada.
Informação Associada: O agente atua ciclicamente varrendo constantemente a base de dados em busca de novos repositórios, e os repositórios em busca de novos artefatos. O Agente Bibliotecário pode determinar a instanciação de um volume maior de Agentes Analisadores

Quadro 02 – *Template* textual do Agente Bibliotecário

[23] O *template* textual é um artefato da linguagem de modelagem de agentes Mas-CommonKads+ utilizado para descrever o comportamento dos agentes.

3.3.2 Agente Analisador

Os componentes mais atuantes da comunidade de agentes do Tohosssou são os chamados Agentes Analisadores. Estes agentes recebem a lista de artefatos de um dado projeto, criada pelo Agente Bibliotecário, juntamente com a informação de que base de conhecimento deve ser utilizada, ou seja, a Ontologia de Aplicação, que será abordada na Seção 3.4.

De acordo com a demanda de trabalho, poderão ser instanciados mais de um Agente Analisador, alocados de forma dinâmica e proporcional ao volume de trabalho, com limite máximo determinado pela capacidade do hardware empregado. Cada instância do Agente Analisador reserva para si um dos artefatos a analisar e comunica aos demais agentes esta reserva, para que não haja competição ou repetição de trabalho, desperdiçando recursos. Ao final das análises, se não há mais artefatos disponíveis para análise e há agentes ociosos, eles decretam o fim de suas atividades, liberando memória e recursos computacionais. Havendo artefatos, estes continuam a analisá-los.

Os Agentes Analisadores foram construídos com base na Ontologia de Domínio, que será descrita na Seção 3.4, e são capazes de reconhecer quaisquer de seus elementos (classes, propriedades, taxonomia) de modo que qualquer nova ontologia que herde as classes da Ontologia de Domínio serão também compreendidas pelos Agentes Analisadores.

Com esta estrutura, o SMA ganha o poder de adaptar-se às necessidades específicas de cada projeto, colaborando com a qualidade do produto final e auxiliando as equipes de desenvolvimento naquilo que elas de fato necessitam. Para isto, basta que uma ontologia apropriada e que derive da Ontologia de Domínio seja incorporada ao Tohossou. O Quadro 03 apresenta o *template* textual do Agente Analisador, descrevendo suas principais características.

Agente Analisador
Objetivo: Analisar artefatos baseados nas ontologias de aplicação.
Parâmetros de Entrada: Artefato e ontologia de aplicação.
Parâmetros de Saída: Relatório da análise.
Condições de Ativação: Mensagem do Agente Bibliotecário.
Condições de Finalização: A análise do artefato foi finalizada.
Informação Associada: O agente só é ativado e mantém-se ativo se houver demanda de análise. Além do artefato, o agente deve ter acesso a uma ou mais ontologias em formato RDF, baseadas na Ontologia de Domínio.

Quadro 03 – *Template* textual do Agente Analisador

3.3.3 Agente de Comunicação

O Agente de Comunicação é responsável pela interação com os participantes do projeto, seja através da interface Web, seja por e-mail, e pela comunicação com outras aplicações via *Web Service*.

Os Agentes mantêm uma troca intensa de mensagens entre si, a maior parte delas transparente para o usuário. Mesmo os relatórios dos Agentes Analisadores têm um nível de complexidade e detalhamento cuja leitura plena poderia gerar mais trabalho para os usuários. Assim, cabe ao Agente de Comunicação filtrar o conteúdo e manter os usuários cientes das ações que estão sendo executadas. Mais ainda, este agente é o interlocutor que relata aos usuários às inconsistências de acessibilidade no código através de e-mail.

O Agente de Comunicação remete relatos distintos a desenvolvedores e gerentes de projeto. Aos primeiros relata que arquivos possuem falhas e quais são estas falhas para que sejam corrigidas. Aos últimos provê resumos informativos e consolidados das alterações e correções realizadas. Além disso, eles remetem mensagens de incentivo aos usuários informando quando não há nenhuma correção a ser feita.

O Agente de Comunicação possui ainda um *Web Service* para possibilitar a integração com outras ferramentas, inclusive de gestão de projetos, como, por exemplo, o DotProject[24]. Através de chamadas e intercâmbio de arquivos em formato XML, é possível, por exemplo, que o Agente de Comunicação solicite a inclusão de uma tarefa para um usuário no sistema de gestão de projetos ou outra ferramenta de colaboração, ou que um projeto seja exportado plenamente da ferramenta de gestão para o Tohossou, facilitando ainda mais a integração.

Uma facilidade não implementada, mas coerente com as tecnologias utilizadas, é a conexão com redes sociais, como facebook[25], twitter[26] ou Google+[27], para que os usuários possam receber as notificações e incentivos de uma forma mais natural e colaborativa. O Quadro 04 apresenta o *template* textual do Agente Analisador, apresentando suas principais características.

[24] http://www.dotproject.net/

[25] http://www.facebook.com/

[26] http://www.twitter.com/

[27] https://plus.google.com/

Agente de Comunicação
Objetivo: Comunicar-se com as entidades externas ao SMA (demais atores)
Parâmetros de Entrada: Relatórios e mensagens no *blackboard.*
Parâmetros de Saída: E-mail enviado.
Condições de Ativação: Mensagem do Agente Analisador.
Condições de Finalização: O e-mail foi enviado ao usuário ou a interação via *Web Service* finalizada
Informação Associada: O agente só é ativado se houver solicitação expressa por mensagem no *blackboard*. Este agente analisa relatórios e o próprio *blackboard* em busca de informações úteis aos usuários para enviá-las via e-mail.

Quadro 04 – *Template* textual do Agente de Comunicação

3.4 Bases de conhecimento

Uma das deficiências detectadas nas ferramentas atuais de validação de conteúdo quanto à acessibilidade é o fato de que as mesmas não incorporam todas as demandas, mas somente aquelas enquadradas em certos padrões, como, por exemplo, o *Section 508* ou o WAI/W3C. Estes padrões são corretos, mas não contemplam todas as faces dos problemas de acessibilidade, e as ferramentas que deles utilizam-se têm a mesma limitação.

Para que o Tohossou não desse continuidade a este paradigma, determinou-se que sua base de conhecimento não deveria ser estática, mas que pudesse ser ampliada com o tempo. Mais ainda, que novas perspectivas e padrões fossem acrescidos sem a necessidade de reconstrução da ferramenta ou de seus agentes. A forma escolhida dentre as disponíveis foi o uso de ontologias.

Para tanto, buscou-se os conceitos de Ontologia de Domínio e Aplicação. As Ontologias de Domínio descrevem o vocabulário e relações de um determinado domínio de conhecimento ou realidade. As Ontologias de Aplicação normalmente são especializações das ontologias de domínio e determinam restrições, aplicação e mesmo instanciação de indivíduos do domínio (GUARINO, 1998).

Os artefatos que serão monitorados pelo Tohossou, seja uma página Web ou código Java, são essencialmente textos, com sintaxes diferentes, codificação diferentes, formatos diferentes, mas sempre um conjunto de caracteres formando palavras e sentenças. Este é o domínio primário do Tohossou: textos.

Contudo, cada linguagem possui especificidades de sintaxe e semântica que lhes são próprias, de modo que para cada linguagem que se deseje codificar, uma Ontologia de Aplicação deve ser criada.

Para a criação de uma ontologia não há um único padrão ou metodologia, e alguns autores têm se debruçado em estudar e escrever acerca de melhores práticas para a criação das mesmas. Contudo, algumas atividades são essenciais a qualquer método que se siga: i) definir as classes; ii) organizá-las hierarquicamente (taxonomicamente - sub/superclasses); iii) definir propriedades e restrições; iv) atribuir os valores das propriedades; e, quando necessário, v) criar instâncias de classes (NOY, N. F. *et al.*, 2001). Para a modelagem da ontologia também não há uma única ferramenta, mas a mais utilizada e referenciada na literatura é o Protégé[28], mantido pela *Stanford University School of Medicine* (NOY, NATALYA F. *et al.*, 2003), que foi também usada na modelagem das ontologias do Tohossou.

Para a criação da Ontologia de Domínio, foi utilizada uma abordagem *top-down*, ou seja, partindo do elemento mais genérico para o mais específico (NOY, N. F. *et al.*, 2001). Desta forma, inferiu-se que um texto escrito, seja um poema ou trecho de código, é essencialmente composto por caracteres, sejam eles letras, números, kanji (ideogramas usados para expressar as palavras no japonês) ou outro qualquer que represente graficamente um elemento da linguagem em uso. Estes, agrupados, formam palavras. As palavras constituem sentenças que, por fim, organizam-se em parágrafos. Esta hierarquia é graficamente apresentada na Figura 11.

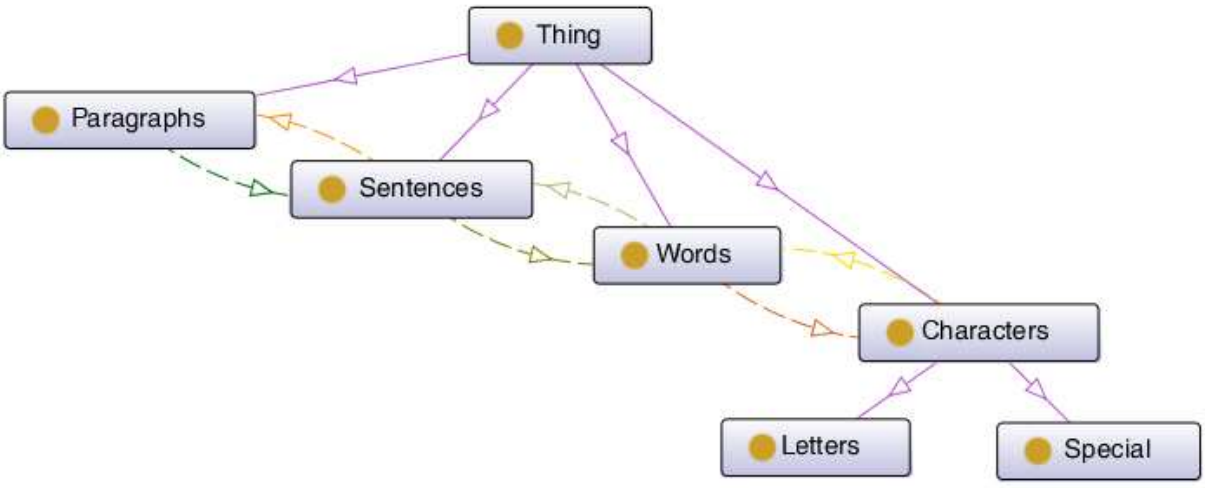


Figura 11 - Ontologia de domínio sobre textos usada pelo Tohossou. As setas representam as relações hierárquicas entre as classes

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

A classe Caracteres (*Characters*) foi desdobrada posteriormente em duas subclasses (Letras e Caracteres Especiais). Esta possibilidade de reavaliação é natural do ambiente de desenvolvimento de software e prevista nas técnicas de desenvolvimento de ontologias (NOY, N. F. *et al.*, 2001).

Não é objetivo do SMA proposto, pelo menos a curto prazo, que os Agentes Analisadores compreendam o significado de cada palavra, mas sim que entendam a inter-relação entre palavras, e em que situação esta relação pode ou não levar a uma condição de falta de acessibilidade. Uma vez que os padrões de acessibilidade estão normalmente relacionados à sintaxe das linguagens, ou à inclusão ou não de parâmetros, ou ainda aos valores de propriedades de certos objetos, se mapeadas estas características em uma ontologia, os Agentes Analisadores podem realizar suas validações com sucesso.

Além das classes, a relação entre elas (*Object Property Hierarchy*) e as propriedades de cada classe (*Data Property Hierarchy*) também foram mapeadas na Ontologia de Domínio como apresentado na Figura 12.

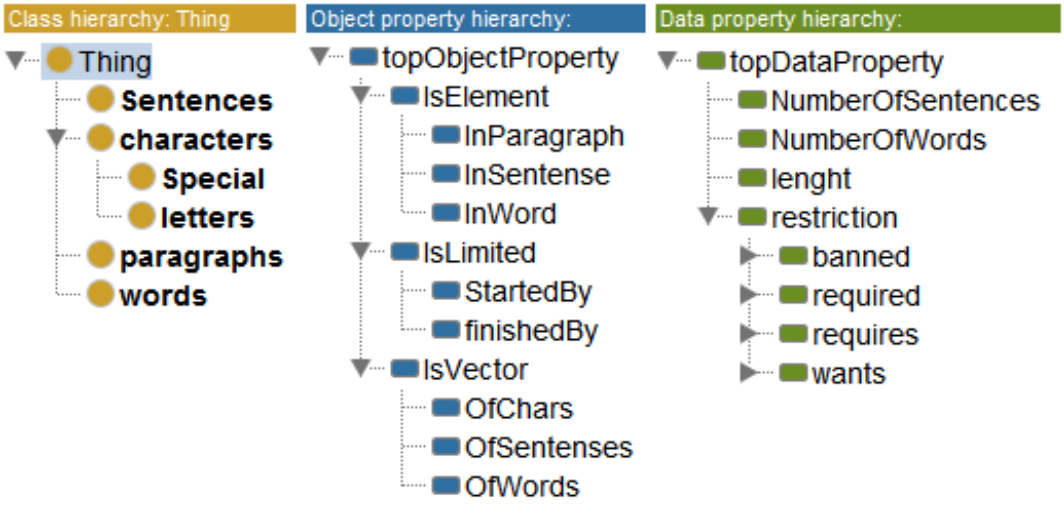


Figura 12 - Classes, propriedades de objetos e dados da Ontologia de Domínio

As propriedades “NumberOfSentences”, “NumberOfWords” e “lenght” são relacionados ao tamanhos (em sentenças, palavras e caracteres). Estes comprimentos podem ser úteis nas análises, como, por exemplo, na detecção de expressões ou palavras muito longas ou muito curtas, e auxilia os agentes em processos iterativos (laços).

Já as propriedades da classe “restriction” são fortemente relacionadas às ações dos agentes de validação. As restrições representam aspectos que devem ser observados pelos Agentes Analisadores. Os aspectos “banned” são situações que não devem estar contidas no

texto em análise. “Required” são aspectos que devem obrigatoriamente estar contidos no artefato. “Requires” deve ser usado quando uma classe ou instância de classe requer a existência de uma dada condição (relação de cardinalidade). E “Wants” refere-se a uma condição dita desejada ou recomendada, cuja existência é positiva, mas a ausência não gera impactos negativos.

As propriedades de objeto “IsElement” representam uma relação entre classes onde uma classe “é elemento” em outra classe, ou “faz parte” de outra classe, como, por exemplo, qualquer instância da classe “word” é elemento numa “sentence” (ou “word” + “IsElement” + “InSentense”).

As propriedades “IsLimited” determinam que palavras, sentenças e parágrafos podem ser limitados no início (“StartedBy”) e no final (“FinishedBy”), por um caractere especial ou um outro elemento.

“IsVector” é a propriedade que determina que uma classe pode ser composta de vetores de outras classes. Esta propriedade visa conciliar o intercâmbio de informações com linguagens que tenham o conceito de vetores bem definido (ex. palavras são vetores de caracteres) e é a propriedade inversa e reflexiva de “IsElement”.

As Ontologias de Aplicação devem ser criadas herdando as classes da Ontologia de Domínio. Para cada tipo de verificação, ou aplicação neste caso, deve ser criada uma nova ontologia que será agregada ao conjunto pré-existente, ampliando a capacidade de análise dos Agentes Analisadores. Para mapear HTML, por exemplo, a ontologia deve definir que uma marcação (*tag*) HTML[29] é classe derivada de “*sentence*”, pois uma marcação pode ter uma ou mais *words*, que se inicia com um “*character*” do tipo “*Special*”, no caso “<”, e termina com outro, neste caso “>” conforme ilustrado na Figura 13.

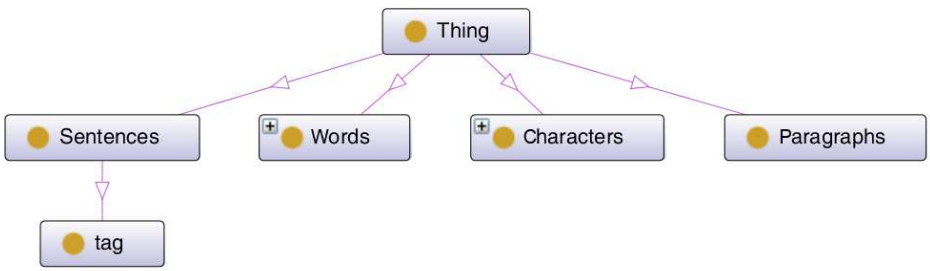


Figura 13 - Exemplo de Ontologia de Aplicação. A classe "tag" (as marcações do HTML) é implementada herdando as características da classe “Sentences”

[29] http://www.w3.org/html/

A necessidade da herança e hierarquia de ontologias é decorrente da preocupação de projetar um SMA dinâmico, que possa evoluir com o tempo, ampliando os domínios de validação e atualizando-se conforme mudem as normas e padrões. Como exemplo, pode-se citar a inclusão de validação de *frameworks* específicos (Yii, Zend[30], CodeIgniter[31], dentre outros) ou leis específicas de cada país, como a lei americana *Section 508*, ou ainda qualquer outra demanda específica que a equipe de desenvolvimento necessite.

3.5 Bases de dados

A estrutura de banco de dados usada pelo Tohossou tem uma grande importância nas atividades executadas pelo sistema. Como o *status* dos projetos, e seus complementos, e o *status* e comunicação dos agentes são armazenados no banco de dados, pode-se garantir a continuidade do trabalho após falhas plenas de funcionamento. Além disso, como o banco também é publicado na Internet, é possível construir locais de *backup* remoto de dados e, principalmente, incluir novos agentes, em plataformas de hardware geograficamente distribuídas, tirando o máximo proveito de processamento em paralelo e garantindo centralização de dados e registros.

Quatro grupos distintos de tabelas podem ser destacados: "*Project*", "*User*", "*Ontology*" e "*Auxiliary Data*", e são visualmente apresentados na Figura 14.

O grupo "*User*" registra em duas tabelas os dados dos usuários e os *tickets* referentes aos seus ingressos. Estas tabelas têm relação direta com o cadastro de novos usuários.

O grupo "*Project*" agrega aos projetos criados pelos líderes, os repositórios a ele vinculados e os artefatos de cada repositório. Projetos e repositórios são dados fornecidos pelo líder ao cadastrar o projeto pela interface Web. A tabela de Artefatos ("*artfacts*") é administrada pelo Agente Bibliotecário, que inclui dados com base nos resultados repassados pelo repositório SVN. A tabela "*Artfactreview*" é atualizada pelo Agente Bibliotecário, com informações sobre versionamento, e pelos Agentes Analisadores, que registram informações sobre a avaliação e sobre a geração de relatórios sobre falhas.

[30] http://www.zend.com/en/

[31] http://codeigniter.com/

O grupo “Ontology”, e sua única tabela, não guarda a ontologia propriamente dita, mas registra todas as ontologias disponíveis ao SMA e a tabela é lida e administrada pelo Agente Bibliotecário.

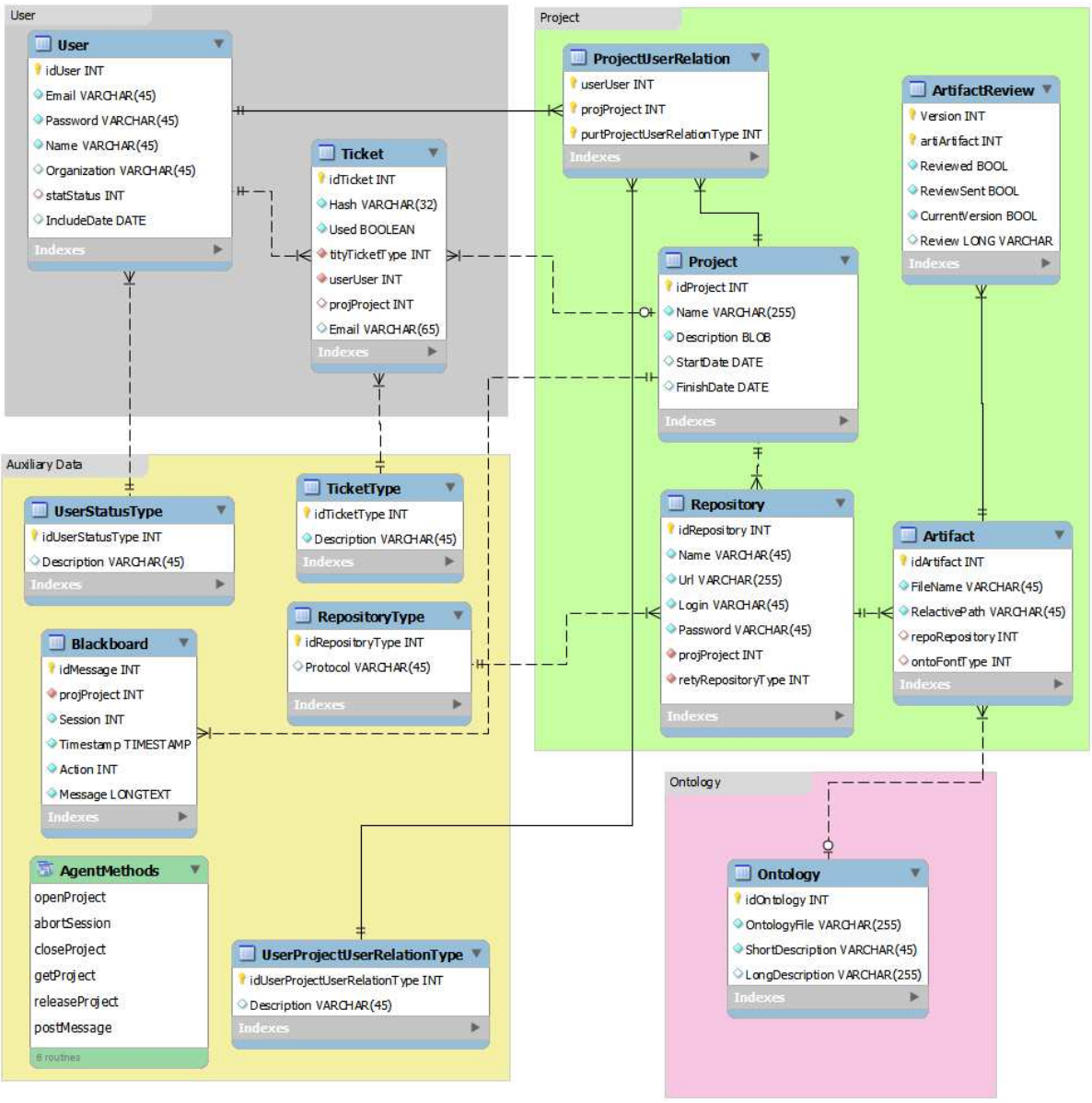


Figura 14 - Representação gráfica do modelo de dados usado pelo Tohossou

O grupo “*Auxiliary Data*” possui tabelas de tipologia de dados e principalmente o *blackboard* usado pelos agentes para comunicação. Ele está vinculado ao projeto de modo que a troca de mensagens entre Agentes Analisadores que estão trabalhando em um projeto não atue erroneamente sobre outros projetos e sobre o trabalho de outros Agentes Analisadores. As mensagens possuem ainda registro temporal, da mesma forma que uma conversa que se realiza ao longo do tempo. Por fim, alguns métodos e consultas muito executadas pelos

agentes foram implementadas diretamente no banco ("*AgentMethods*"). Essa implementação torna a execução mais rápida e diminui drasticamente o tráfego de dados entre os agentes e o SGBD[32].

3.6 Tecnologias e técnicas utilizadas e detalhes da implementação

O Tohossou foi implementado em dois modos de funcionamento distintos: interface interativa com usuário e agentes autônomos. Em ambas, a codificação foi feita como a linguagem PHP, com base nos preceitos já indicados. Um dos motivos que justificou a escolha foi o fato de ser uma linguagem orientada a objetos, similar em filosofia e sintaxe a outras linguagens de programação, como o Java[33], abrindo a possibilidade para que todo o Tohossou possa ser reimplementado em outra linguagem, caso seja necessário.

Sobre a linguagem PHP, foi utilizado o framework Yii. Sendo um framework de desenvolvimento de interface, o Yii permitiu a rápida implementação do Tohossou, agregando as funcionalidades necessárias para acesso ao banco de dados MySQL. Este framework não entra em conflito com as bibliotecas de acesso a repositórios SVN do PHP e permite agregar outras bibliotecas. O Yii possui recursos para desenvolvimento de *Web Services*, o que permitiu a implementação da interface adicional no agente de comunicação sem trazer dificuldades adicionais de codificação.

Em adição a esta estrutura, foi usada a biblioteca da RAP (RDF API for PHP[34]) versão 0.9.6. Tal biblioteca disponibiliza funções e classes para leitura de arquivos de ontologias em formato RDF, podendo-se "ler" as ontologias de modo estrutural (OLDAKOWSKI *et al.*, 2005) e realizar consultas sobre as mesmas na forma de cláusulas SPARQL (PRUD'HOMMEAUX; SEABORNE, 2006). Na versão usada, a RAP não dava suporte a arquivos de formato OWL, bem mais populares. No entanto, os recursos disponibilizados justificaram a falta de compatibilidade, além de que existem várias ferramentas que podem ser usadas para converter de um formato para outro sem perda de informações (NOY, NATALYA F. *et al.*, 2003).

[32] Acrônimo para Sistema de Gerenciamento de Banco de Dados.

[33] http://www.java.com/pt_BR/

[34] http://sourceforge.net/projects/rdfapi-php/

Fazendo uso do modelo MVC (*Model-View-Controller* (DEACON, 2005)), o Yii disponibiliza a infraestrutura para funcionamento de toda a aplicação, principalmente ao intermediar a comunicação com o banco de dados e gerir a criação e atualização dinâmica da interface Web.

Após ser criado na interface, cada Agente Bibliotecário, se houver mais de um, fica responsável por ler os projetos e todos seus repositórios. O projeto em análise torna-se indisponível aos demais agentes. O Agente Bibliotecário usa a biblioteca SVN do PHP para acessar cada repositório do seu projeto, vasculhando-os e recuperando informações sobre artefatos e suas versões. A indisponibilidade aos demais agentes dá-se porque o protocolo SVN é bastante prolixo, incluindo em suas mensagens numerosos dados não opcionais que são transmitidos mesmo quando não necessários (OTTE, 2009) e um repositório, por menor que seja seu conteúdo, pode levar alguns minutos para ser analisado.

Os Agentes Analisadores procuram no *blackboard* por sinalizações de desbloqueio de projeto e artefatos gerados pelos Agentes Bibliotecários e inicia suas análises.

Os Agentes Analisadores fazem uso da biblioteca RAP para ler a estrutura das Ontologias de Aplicação, contidas em um ou mais arquivos RDF, e recupera as classes e os itens a analisar fazendo uso de cláusulas SPARQL[35]. O *OntModel*, indicado no pseudocódigo do quadro 5, é uma classe codificada dentro do agente Analisador. Esta classe é responsável por ler o arquivo que contém uma ontologia, verificar a árvore de classes definida no arquivo, recuperar cada propriedade de cada classe, recuperar os indivíduos definidos na ontologia, gerar as expressões regulares e fazer a revisão (*parsing*) de um arquivo fonte.

```
Recuperar do banco de dados as informações de repositórios do projeto;
PARA cada repositório
    Recuperar informações de cada arquivo que tenha sido marcado para analise, por
      agentes bibliotecários;
    PARA cada arquivo recuperado
      Recuperar ontologia associada ao arquivo;
      Armazenar ontologia em OntModel;
      Recuperar código fonte do arquivo;
      Solicitar ao OntModel que faça análise de arquivo e devolva críticas;
      Salvar críticas junto às informações do arquivo;
      Remover marcação de necessidade de análise por agentes Analisadores;
      Marcar arquivo como sendo necessário enviar críticas sobre o mesmo;
    FIM PARA;
FIM PARA;
```

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

Quadro 05 – ***Pseudocódigo de execução de agente Analisador***

As ontologias são manipuladas através da biblioteca RAP (RDF API for PhP). Com esta biblioteca é possível realizar consultas SPARQL, ou “ler" a estrutura do arquivo. A classe *OntModel* consulta os arquivos de ontologia usando as duas técnicas.

A fim de recuperar as informações de classes e propriedades de classes para criação de expressões regulares, a classe *OntModel* realiza consultas SPARQL (como no trecho de código contido no quadro 6). Estando montadas as expressões regulares, uma *OntMode*l pode aplicar cada uma delas a um arquivo, detectando as ocorrências das mesmas e agrupando as mensagens relativas a presença ou ausência das expressões, de acordo com o definido na ontologia.

```
private function getClasses(){
    $query=$this->_localPrefix.
       "SELECT ?subject WHERE { ?subject rdfs:subClassOf tc:Sentences . }";
    $result = $this->_model->sparqlQuery($query);
    foreach($result as $class){
       ...
       $query=$this->_localPrefix.
            " SELECT DISTINCT ?property WHERE {?s a local:".
            $className.";?property ?o }";
       $props=$this->_model->sparqlQuery($query);
       foreach($props as $prop){
          $propName=substr($prop["?property"]->uri,
             strpos($prop["?property"]->uri,'#')+1);
          if($propName=="type"){
             continue;
          }
          $this->_classes[$className][]=$propName;
       }
    }
}
```

Quadro 06 – ***Pseudocódigo de execução de agente Analisador***

Com o uso de expressões regulares (FRIEDL, 2006) que representam as possíveis não-conformidades, os agentes analisadores lêem os arquivos fontes nos repositórios, através de *streaming*, ou seja, a leitura e avaliação do artefato é feita durante o *download*, não sendo necessária armazenamento de cópia física do arquivo, exceto em *cache* temporário, o que

favorece a leitura de arquivos de grande porte e restringe o espaço de armazenamento ao *cache* durante a leitura (AHRENS *et al.*, 2001). Para cada arquivo, o agente cria um relatório de verificação que é analisado pelo Agente de Comunicação e remetido, se necessário, aos usuários.

Os agentes podem ser criados tanto por chamadas diretas (solicitação ao servidor Web) feita pelo usuário ou por outros agentes, especialmente pelo Agente Bibliotecário, que dimensiona a carga de trabalho a ser cumprida, ou por agendamento, por métodos como o uso do *cron*[36] (REZNICK, 1993).

3.6.1 *Blackboard* e comunicação entre agentes

Devido ser um sistema multi-agentes, o Tohossou necessita de um ambiente que permita a comunicação entre os agentes que o compõem. Dentre as diversas possibilidades de implementação, foi utilizado *blackboard*, uma área de memória comum onde os agentes podem ler e publicar informações antes de agir ou para fundamentar suas ações.

O *blackboard* foi implementado em um banco de dados MySQL em uma tabela (conforme visto na figura 14) com uma estrutura compreensível a todos os agentes e passível de ampliação para necessidades futuras. Cada registro no banco é composto de um identificador único das mensagens (idMessage); um identificação de qual projeto se refere a mensagem (poderá ser nulo de a mensagem não referir-se a nenhum projeto – idProject); um identificação ordinal de qual sessão de verificação do projeto (Session); o momento de escrita da mensagem do *blackboard* (*Timestamp*) e a ação (Action) que pode ou está sendo executado pelo agente.

No entanto, os agentes não fazem leitura e escrita diretamente na área comum, mas sim através de primitivas escritas no próprio banco na forma de UDF (*User Defined Function*). Com as UDF o tratamento de preocupações típicas de ambientes muti-agentes, como a concorrência no acesso aos recursos, por exemplo, ficam a cargo do Baco de dados (ORACLE CORPORATION, 2011). Adotou-se algumas primitivas como: *openProject* (Abre um projeto que não tenha sido analisado, que esteja fechado ou que sua última sessão de análise tenha sido abortada); *closeProject* (Finaliza uma sessão de análise de um projeto, indicando que todo o processo foi bem sucedido); *getProject* (Verifica se há algum projeto

[36] http://www.linux.org/article/view/automation-of-tasks

disponível para a fase de análise do agente. Se houver, é posta, no blackboard, uma mensagem relativa à fase de análise e é devolvida ao agente a identificação do projeto disponível), dentre outras. Exemplos destas primitivas estão nos quadros 7 e 8.

GetProject
... **SE o tipo de agente for bibliotecário ENTÃO** **Buscar apenas um projeto qualquer que tenha sido aberto, mas não tenha sofrido nenhum tipo de análise;** **SE Existir ENTÃO** **Postar mensagem no blackboard que o projeto está em análise por agente bibliotecário;** **FIM SE;** **...** **(Fazer verificação para os outros tipos de agentes)** **...** **FIM SE;** **RETORNAR Projeto recuperado ou Nada (se nenhum projeto estiver disponível).**

Quadro 07 – ***Pseudo-código da Primitiva GetProject***

ReleaseProject
... **Recuperar informações da sessão do projeto informado pelo agente;** **Verificar o tipo de ação em curso** **Postar mensagem de encerramento de atividades do tipo de agente na sessão;** **SE o tipo de agente for "Mensageiro" ENTÃO** **Solicitar fechamento da sessão do projeto;** **SE não conseguir fechar o projeto ENTÃO** **Abortar sessão;** **FIM SE;** **FIM SE;** **RETORNAR identificação do projeto.**

Quadro 08 – ***Pseudo-código da Primitiva ReleaseProject***

4 Considerações Finais e Trabalhos Futuros

Os métodos tradicionais de validação de acessibilidade ocorrem *a posteriori*, exigindo refatoração e, consequentemente, retrabalho por parte da equipe de desenvolvimento. Além disso, muitas equipes trabalham hoje de forma colaborativa e geograficamente distribuídas, apoiadas em ferramentas de controle de versões de código, o que dificulta a validação de acessibilidade em partes de código de diferentes desenvolvedores. Tudo isso, aliado à falta de conhecimento técnico por parte dos desenvolvedores acerca das normas de acessibilidade, visto que este conhecimento não faz parte do currículo da maioria dos cursos de ciência da computação, torna complexa a tarefa de garantir acessibilidade em projetos de aplicações para Web.

Como uma alternativa a este cenário que culmina na falta de acessibilidade nas aplicações e conteúdo para Web, este trabalho propôs um SMA, com uso de ontologias como base de conhecimento, que monitora o trabalho da equipe de desenvolvimento, local ou distribuída, alertando-a, em tempo real, acerca de não-conformidades quanto à acessibilidade.

O Tohosssou encontra-se em fase de homologação e publicação do serviço para a comunidade. Nesta etapa, considerada beta, serão incorporadas melhorias sugeridas pelos usuários e correções de *bugs*, em especial os decorrentes de uso com uma carga maior de tarefas.

A conjunção das tecnologias descritas apresentou-se adequada aos resultados almejados. O uso de ontologias como base de conhecimento para os agentes, que já foi relatado como positivo em pesquisas recentes, comprovou-se com o Tohossou. Os mecanismos de inferência e o nível de detalhamento oferecido pelas ontologias são adequados para esta e outras aplicações, além de naturalmente permitirem a expansão contínua do conhecimento sem a necessidade de alteração dos agentes.

Foram realizados testes de alteração e incorporação de regras distintas nas ontologias e, uma vez seguidas as heranças e a estrutura da Ontologia de Domínio, o comportamento dos Agentes Analisadores mudou, em consonância com a base de conhecimento, sem alteração do desempenho e da qualidade dos resultados.

A escolha natural de uma linguagem de implementação para uma ferramenta desta amplitude normalmente seria o Java, mas a opção pelo uso de PHP e diversos pacotes de apoio (como, por exemplo, para acessar os repositórios e as ontologias) e *frameworks* (neste caso, o Yii), mostrou-se acertada. Além da facilidade de programação, depuração e incorporação de novas funcionalidades, o PHP é nativo para a Internet (e seus protocolos

principais como o FTP), incorporando muitas das características que se busca para uma ferramenta dentro do paradigma do *Cloud Computing*. Nenhuma funcionalidade mapeada nos requisitos do sistema deixou de ser implementada pela ausência de suporte da linguagem.

Optou-se pelo SVN como repositório nativo do Tohossou dada a sua ampla base de usuários e pelo controle efetivo que realiza sobre o repositório, o que não seria possível com um repositório baseado em FTP, por exemplo, onde a gestão de versões deveria estar sob a tutela do Tohossou. Contudo, usar outros repositórios não representaria mudanças significativas.

A qualidade dos resultados das avaliações de não-conformidade do Tohossou não diverge de ferramentas tradicionais, sem, contudo, ser necessário o final do projeto para seu uso, evitando retrabalho da equipe de desenvolvimento. Em projetos já concluídos, o benefício está relacionado ao monitoramento contínuo do repositório, caso haja mudanças.

Como trabalhos futuros, planeja-se incorporar, aos Agentes de Comunicação, a possibilidade de comunicar-se com as redes sociais e outras ferramentas de comunicação e colaboração em ascensão.

Pretende-se também construir um volume de ontologias de aplicação que permita validação de um conjunto maior de artefatos. Existe hoje um sem número de *frameworks* para a produção de conteúdo para web, e sua grande maioria não dá suporte explícito à acessibilidade, de modo que a validação prévia proposta pelo Tohossou permitiria minimizar esta falta de suporte.

Além disso, as múltiplas plataformas de hardware, em especial os dispositivos móveis, possuem características próprias quanto à acessibilidade que impactam nas técnicas de desenvolvimento e, através de ontologias de aplicação específicas, estas plataformas podem ser beneficiadas pelo Tohossou. O mapeamento destas características em ontologias é uma meta concreta e que pode render bons frutos.

Por fim, pode-se preparar o sistema para aplicação de técnicas de reconhecimento de linguagem natural, possibilitando que a análise dos agentes incorpore todo o potencial da semântica.

5 Referências

AHMED, M.; AHMAD, M. S.; MOHD YUSOFF, M. Z. A Review and Development of Agent Communication Language. electronic Journal of Computer Science and Information Technology, v. 1, n. 1. Retrieved July 31, 2012, from http://ejcsit.uniten.edu.my/index.php/ejcsit/article/view/3, 2011.

AHRENS, J.; BRISLAWN, K.; MARTIN, K. et al. Large-scale data visualization using parallel data streaming. IEEE Computer Graphics and Applications, v. 21, n. 4, p. 34 –41. doi: 10.1109/38.933522, 2001.

ANTONIOU, G.; HARMELEN, FRANK VAN. Web Ontology Language: OWL. In: Steffen Staab; D. Rudi Studer (Eds.); Handbook on Ontologies, International Handbooks on Information Systems.. p.91–110. Springer Berlin Heidelberg. Retrieved July 30, 2012, from http://www.springerlink.com/content/w378kq77t5504546/abstract/, 2009.

AQUINO, F.; CRISTINA, L. E-ACCESIBILIDAD Y USABILIDAD DE CONTENIDOS DIGITALES. POR UNA SOCIEDAD DE LA INFORMACIÓN Y EL CONOCIMIENTO NO EXCLUYENTE. Riunet. Retrieved July 29, 2012, from http://riunet.upv.es/handle/10251/4330, 2009.

ARMBRUST, M.; FOX, A.; GRIFFITH, R. et al. A view of cloud computing. Commun. ACM, v. 53, n. 4, p. 50–58. doi: 10.1145/1721654.1721672, 2010.

ARTERO, A. O. Inteligência Artificial - Teórica e Prática. 1st ed. Livraria das Física. Retrieved June 22, 2012, , 2009.

BATISTA, J. C. Os efeitos das políticas industriais para o setor de produtos eletrônicos do Brasil. Revista de Economia Política, v. 30, n. 1, p. 112–123. doi: 10.1590/S0101-31572010000100007, 2010.

BENYON, D.; TURNER, D. P.; TURNER, D. S. Designing Interactive Systems: People, Activities, Contexts, Technologies. Addison-Wesley, 2005.

BERNARDES, L. C. G.; MAIOR, I. M. M. DE L.; SPEZIA, C. H.; ARAUJO, T. C. C. F. DE. Persons with disabilities and Brazilian healthcare policies: thoughts about bioethics. Ciência & Saúde Coletiva, v. 14, n. 1, p. 31–38. doi: 10.1590/S1413-81232009000100008, 2009.

BERNERS-LEE, T.; HENDLER, J. Publishing on the semantic web. Nature, v. 410, n. 6832, p. 1023–1024. Retrieved August 7, 2012, , 2001.

BRADNER, E.; MARK, G.; HERTEL, T. D. Effects of Team Size on Participation, Awareness, and Technology Choice in Geographically Distributed Teams. Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 8 -

Volume 8. Anais... , HICSS ’03.. p.271.1–. Washington, DC, USA: IEEE Computer Society. Retrieved June 22, 2012, from http://dl.acm.org/citation.cfm?id=820755.821737, 2003.

BRASIL. Constituição: República Federativa do Brasil. ,1998.

BRASIL. 10.098, de 19 de dezembro de 2000.Diário Oficial da União. v. 244, p.2, 2000.

BRUNET, P.; FEIGENBAUM, B. A.; HARRIS, K. et al. Accessibility requirements for systems design to accommodate users with vision impairments. IBM Systems Journal, v. 44, n. 3, p. 445 –466. doi: 10.1147/sj.443.0445, 2005.

CANDEIAS, A. A.; ALMEIDA, L. S.; ROAZZI, A.; PRIMI, R. Inteligência: Definição e medida na confluência de multiplas concepções. Retrieved July 31, 2012, from http://www.rdpc.uevora.pt/handle/10174/1827, 2008.

CASEY, V.; RICHARDSON, I. Implementation of Global Software Development: a structured approach. Software Process: Improvement and Practice, v. 14, n. 5, p. 247–262. doi: 10.1002/spip.422, 2009.

CGPID, C. G. DO P. DE I. D. Plano Nacional de Banda Larga. . Retrieved from http://www.mc.gov.br/publicacoes/doc_download/418-documento-base-do-programa-nacional-de-banda-larga, 2010.

CHISHOLM, W. A.; HENRY, S. L. Interdependent components of web accessibility. Proceedings of the 2005 International Cross-Disciplinary Workshop on Web Accessibility (W4A). Anais... , W4A ’05.. p.31–37. New York, NY, USA: ACM. doi: 10.1145/1061811.1061818, 2005.

CHISHOLM, W.; VANDERHEIDEN, G.; JACOBS, I. Web content accessibility guidelines 1.0. interactions, v. 8, n. 4, p. 35–54. doi: 10.1145/379537.379550, 2001.

CONFORTO, D.; SANTAROSA, L. M. C. ACESSIBILIDADE À WEB : INTERNET PARA TODOS. Revista de Informática na Educação: Teoria, Prática, v. 5, n. 2, p. 87–102, 2002.

CORDE, C. N. PARA A I. DA P. P. DE D. Normas ABNT. . Retrieved June 15, 2010, from http://portal.mj.gov.br/corde/normas_abnt.asp, 2004.

DAVIES, J.; FENSEL, DIETER; HARMELEN, FRANK VAN (EDS.). Front Matter. Towards the Semantic Web. p.i–xxi. John Wiley & Sons, Ltd. Retrieved June 22, 2012, from http://onlinelibrary.wiley.com/doi/10.1002/0470858060.fmatter/summary, 2003.

DEACON, J. Model-View-Controller (MVC) Architecture. JOHN DEACON Computer Systems Development, Consulting & Training. Retrieved August 1, 2012, from http://wiki.pureweber.com/uploads/Ebooks/%E8%BD%AF%E4%BB%B6%E8%AE%BE%E

8%AE%A1%E8%AE%BE%E8%AE%A1%E6%A8%A1%E5%BC%8F%E7%B3%BB%E7%BB%9F%E6%9E%B6%E6%9E%84/MVC.pdf, 2005.

DIXIT, S.; OJANPERA, T.; NEE, R.; PRASAD, R. Introduction to Globalization of Mobile and Wireless Communications: Today and in 2020. In: R. Prasad; S. Dixit; R. van Nee; T. Ojanpera (Eds.); Globalization of Mobile and Wireless Communications, Signals and Communication Technology.. p.1–8. Springer Netherlands. Retrieved July 23, 2012, from http://www.springerlink.com/content/th58162xp6u35385/abstract/, 2011.

FENSEL, D.; VAN HARMELEN, F.; HORROCKS, I.; MCGUINNESS, D. L.; PATEL-SCHNEIDER, P. F. OIL: an ontology infrastructure for the Semantic Web. IEEE Intelligent Systems, v. 16, n. 2, p. 38 – 45. doi: 10.1109/5254.920598, 2001.

FENSEL, DIETER; FACCA, F. M.; SIMPERL, E. et al. Semantic Web. Semantic Web Services. p.87–104. Springer Berlin Heidelberg. Retrieved July 29, 2012, from http://www.springerlink.com/content/h309rv470226h5x8/abstract/, 2011.

FERBER, J. Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence. 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

FONTES, L. M. DE O.; NETO, F. M. M.; PONTES, A. Á. A.; CAMPOS, G. A. DE L. An agent-based architecture for supporting the workgroups creation and the detection of out-of-context conversation on problem-based learning in virtual learning environments. Proceedings of the 2011 ACM Symposium on Applied Computing. Anais... , SAC '11.. p.1175–1180. New York, NY, USA: ACM. doi: 10.1145/1982185.1982444, 2011.

FRANKLIN, S.; GRAESSER, A. Is It an agent, or just a program?: A taxonomy for autonomous agents. In: J. Müller; M. Wooldridge; N. Jennings (Eds.); Intelligent Agents III Agent Theories, Architectures, and Languages, Lecture Notes in Computer Science.. v. 1193, p.21–35. Springer Berlin / Heidelberg. Retrieved July 31, 2012, from http://www.springerlink.com/content/w5m51l674402vr07/abstract/, 1997.

FRATERNALI, P.; ROSSI, G.; S&#X0E1 ANDNCHEZ-FIGUEROA, F. Rich Internet Applications. IEEE Internet Computing, v. 14, n. 3, p. 9 –12. doi: 10.1109/MIC.2010.76, 2010.

FRIEDL, J. Mastering Regular Expressions. O'Reilly Media, Inc., 2006.

GARCÍA-SÁNCHEZ, F.; VALENCIA-GARCÍA, R.; MARTÍNEZ-BÉJAR, R.; FERNÁNDEZ-BREIS, J. T. An ontology, intelligent agent-based framework for the provision of semantic web services. Expert Systems with Applications, v. 36, n. 2, Part 2, p. 3167–3187. doi: 10.1016/j.eswa.2008.01.037, 2009.

GUARINO, N. Formal Ontology in Information Systems: Proceedings of the First International Conference (FOIS'98), June 6-8, Trento, Italy. IOS Press, 1998.

GUNASEKARAN, V.; HARMANTZIS, F. C. Towards a Wi-Fi ecosystem: Technology integration and emerging service models. Telecommunications Policy, v. 32, n. 3–4, p. 163–181. doi: 10.1016/j.telpol.2008.01.002, 2008.

HADZIC, M.; WONGTHONGTHAM, PORNPIT; CHANG, ELIZABETH; DILLON, T. Ontology-Based Multi-Agent Systems. Springer, 2009.

HENDLER, J. Agents and the Semantic Web. Intelligent Systems, IEEE, v. 16, n. 2, p. 30 – 37. doi: 10.1109/5254.920597, 2001.

HOGAN, A.; HARTH, A.; UMBRICH, J. et al. Searching and browsing Linked Data with SWSE: The Semantic Web Search Engine. Web Semantics: Science, Services and Agents on the World Wide Web, v. 9, n. 4, p. 365–401. doi: 10.1016/j.websem.2011.06.004, 2011.

HORRIDGE, M. A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools. . The University Of Manchester. Retrieved from http://www.bax.com.br/teaching/courses/gestaodeconteudos/Protege%20OWL%20Tutorial%20P4_v1_2.pdf, 2009, March 13.

HÜBNER, J. F. Um Modelo de Reorganização de Sistemas Multiagentes. São Paulo: Escola Politécnica da Universidade de São Paulo. Retrieved May 1, 2012, from http://www.das.ufsc.br/~jomi/pubs/Hubner-tese.pdf, 2003.

ITU, I. T. U. The world in 2010: ICT facts and figures. International Telecommunication Union, 2010.

JAEGER, P. T. Assessing Section 508 compliance on federal e-government Web sites: A multi-method, user-centered evaluation of accessibility for persons with disabilities. Government Information Quarterly, v. 23, n. 2, p. 169–190. doi: 10.1016/j.giq.2006.03.002, 2006.

LACLAVÍK, M.; BALOGH, Z.; BABÍK, M.; HLUCHÝ, L. AgentOWL: Semantic Knowledge Model and Agent Architecture. COMPUTING AND INFORMATICS, v. 25, n. 5, p. 421–439. Retrieved July 31, 2012, , 2006.

LANUBILE, F.; EBERT, C.; PRIKLADNICKI, R.; VIZCAINO, A. Collaboration Tools for Global Software Engineering. IEEE Software, v. 27, n. 2, p. 52 –55. doi: 10.1109/MS.2010.39, 2010.

LEINER, B. M.; CERF, V. G.; CLARK, D. D. et al. The past and future history of the Internet. Commun. ACM, v. 40, n. 2, p. 102–108. doi: 10.1145/253671.253741, 1997.

LESSER, V. R. Cooperative multiagent systems: a personal view of the state of the art. IEEE Transactions on Knowledge and Data Engineering, v. 11, n. 1, p. 133 –142. doi: 10.1109/69.755622, 1999.

LI, L.; WU, B.; YANG, Y. Agent-based ontology integration for ontology-based applications. Proceedings of the 2005 Australasian Ontology Workshop - Volume 58. Anais... , AOW '05.. p.53–59. Darlinghurst, Australia, Australia: Australian Computer Society, Inc. Retrieved June 22, 2012, from http://dl.acm.org/citation.cfm?id=1151936.1151943, 2005.

LI, S.; HU, F. Communication between the RoboCup Agents Based on the Blackboard Model and Observer Pattern. 5th International Conference on Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. Anais... p.1 –5. doi: 10.1109/WICOM.2009.5304011, 2009.

LÓPEZ, J.; PASCUAL, A.; MASIP, L.; GRANOLLERS, T.; CARDET, X. Influence of Web Content Management Systems in Web Content Accessibility. In: P. Campos; N. Graham; J. Jorge; et al. (Eds.); Human-Computer Interaction – INTERACT 2011, Lecture Notes in Computer Science.. v. 6949, p.548–551. Springer Berlin / Heidelberg. Retrieved July 29, 2012, from http://www.springerlink.com/content/g8347867900801v4/abstract/, 2011.

LUDI, S. Introducing Accessibility Requirements through External Stakeholder Utilization in an Undergraduate Requirements Engineering Course. Proceedings of the 29th international conference on Software Engineering. Anais... , ICSE '07.. p.736–743. Washington, DC, USA: IEEE Computer Society. doi: 10.1109/ICSE.2007.46, 2007.

LUQUE CENTENO, V.; DELGADO KLOOS, C.; ARIAS FISTEUS, J.; ÁLVAREZ ÁLVAREZ, L. Web Accessibility Evaluation Tools: A Survey and Some Improvements. Electronic Notes in Theoretical Computer Science, v. 157, n. 2, p. 87–100. doi: 10.1016/j.entcs.2005.12.048, 2006.

MASUWA-MORGAN, K.R; BURRELL, P. Justification of the need for an ontology for accessibility requirements (Theoretic framework). Interacting with Computers, v. 16, n. 3, p. 523–555. doi: 10.1016/j.intcom.2004.04.001, 2004.

MASUWA-MORGAN, KRISTINA RUNGANO. Introducing AccessOnto: Ontology for Accessibility Requirements Specification. Ontologies in Interactive Systems, International Workshop on. Anais... p.33–38. Los Alamitos, CA, USA: IEEE Computer Society. doi: http://doi.ieeecomputersociety.org/10.1109/ONTORACT.2008.18, 2008.

MATTOS, F. A. M. DE; CHAGAS, G. J. DO N. Challenges for digital inclusion in Brazil. Perspectivas em Ciência da Informação, v. 13, n. 1, p. 67–94. doi: 10.1590/S1413-99362008000100006, 2008.

MCGOOKIN, D.; BREWSTER, S.; JIANG, W. Investigating touchscreen accessibility for people with visual impairments. Proceedings of the 5th Nordic conference on Human-computer interaction: building bridges. Anais... , NordiCHI '08.. p.298–307. New York, NY, USA: ACM. doi: 10.1145/1463160.1463193, 2008.

MOHANTY, S. A new architecture for 3G and WLAN integration and inter-system handover management. Wireless Networks, v. 12, n. 6, p. 733–745. doi: 10.1007/s11276-006-6055-y, 2006.

MORAES, F. D. DE. Inclusão digital em telecentros e Lan Houses : estudo sobre as novas dinâmicas espaciais decorrentes dos centros públicos de acesso à Internet na região norte de Porto Alegre. . Retrieved July 23, 2012, from http://www.lume.ufrgs.br/handle/10183/25948, 2009.

MORAIS II, M. J. DE O. MAS-CommonKADS+: Uma Extensão à Metodologia Mas-CommonKADS para Suporte ao Processo Detalhado de Sitemas Multiagentes Racionais. Fortaleza-CE: Universidade Estadual do Ceará - UECE, 2010.

NOY, N. F.; MCGUINNESS, D. L.; OTHERS. Ontology development 101: A guide to creating your first ontology. Stanford knowledge systems laboratory technical report KSL-01-05 and Stanford medical informatics technical report SMI-2001-0880. Retrieved July 31, 2012, from http://er.uni-koblenz.de/IFI/AGStaab/Teaching/SS09/sw09/Ontology101.pdf, 2001.

NOY, NATALYA F.; CRUBÉZY, M.; FERGERSON, R. W. et al. Protégé-2000: An Open-Source Ontology-Development and Knowledge-Acquisition Environment. AMIA Annual Symposium Proceedings, v. 2003, p. 953. Retrieved August 1, 2012, , 2003.

OLDAKOWSKI, R.; BIZER, C.; WESTPHAL, D. RAP: RDF API for PHP. Proceedings of the 1st Workshop on Scripting for the Semantic Web, 2nd European Semantic Web Conference (ESWC2005). Anais... . Retrieved August 1, 2012, from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.59.7606&rep=rep1&type=pdf, 2005.

ORACLE CORPORATION. MySQL :: MySQL 5.6 Reference Manual :: E.1 Restrictions on Stored Programs. . Retrieved October 10, 2011, from http:dev.mysql.comdocrefman5.6en/stored-program-restrictions.html, 2011.

OREILLY, T. What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. SSRN eLibrary. Retrieved July 23, 2012, from http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1008839, 2007.

OTTE, S. Version Control Systems. Computer Systems and Telematics Institute of Computer Science Freie Universität Berlin, Germany. Retrieved August 1, 2012, from http://cst.mi.fu-berlin.de/teaching/WS0809/19565-PS-TI/reports/otte09version.pdf, 2009.

PLESSERS, P.; CASTELEYN, S.; YESILADA, Y. et al. Accessibility: a Web engineering approach. Proceedings of the 14th international conference on World Wide Web. Anais... , WWW '05.. p.353–362. New York, NY, USA: ACM. doi: 10.1145/1060745.1060799, 2005.

PONTES, A. Á. Uma Arquitetura de Agentes para Suporte a Colaboração na Aprendizagem Baseada em Problemas em Ambientes Virtuais de Aprendizagem. Mossoró-RN: Universidade Federal Rural do Semi-árido. Retrieved from Portugues, 2010.

PRUD'HOMMEAUX, E.; SEABORNE, A. SPARQL Query Language for RDF. Retrieved August 1, 2012, from http://www.w3.org/TR/rdf-sparql-query/, 2006.

PUDER, A. Extending desktop applications to the web. Proceedings of the 2004 international symposium on Information and communication technologies. Anais... , ISICT '04.. p.8–13. Trinity College Dublin. Retrieved July 23, 2012, from http://dl.acm.org/citation.cfm?id=1071509.1071512, 2004.

RAMAN, T. V. Cloud computing and equal access for all. Proceedings of the 2008 international cross-disciplinary conference on Web accessibility (W4A). Anais... , W4A '08.. p.1–4. New York, NY, USA: ACM. doi: 10.1145/1368044.1368046, 2008.

REID, L. G.; SNOW-WEAVER, A. WCAG 2.0: a web accessibility standard for the evolving web. Proceedings of the 2008 international cross-disciplinary conference on Web accessibility (W4A). Anais... , W4A '08.. p.109–115. New York, NY, USA: ACM. doi: 10.1145/1368044.1368069, 2008.

REZENDE, S. O. Sistemas inteligentes: fundamentos e aplicações. Editora Manole Ltda, 2003.

REZNICK, L. Using cron and crontab. Sys Admin, v. 2, n. 4, p. 29–32. Retrieved August 1, 2012, , 1993.

RIBERA, M.; PORRAS, M.; BOLDU, M. et al. Web Content Accessibility Guidelines 2.0: A further step towards accessible digital information. Program: electronic library and information systems, v. 43, n. 4, p. 392–406. doi: 10.1108/00330330910998048, 2009.

RITCHIE, H.; BLANCK, P. The promise of the Internet for disability: a study of on-line services and web site accessibility at Centers for Independent Living. Behavioral Sciences & the Law, v. 21, n. 1, p. 5–26. doi: 10.1002/bsl.520, 2003.

RUSSELL, S. J.; NORVIG, P. Artificial Intelligence: A Modern Approach. Prentice Hall, 2010.

SABATER, J.; SIERRA, C. Reputation and social network analysis in multi-agent systems. Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1. Anais... , AAMAS '02.. p.475–482. New York, NY, USA: ACM. doi: 10.1145/544741.544854, 2002.

SALDANHA, P. G.; EL HAJJI, M. Telecentro: instrumento tecnológico de fortalecimento das atividades humanas. Ciberlegenda, v. 0, n. 16. Retrieved July 23, 2012, from http://www.uff.br/ciberlegenda/ojs/index.php/revista/article/view/204, 2011.

SHADBOLT, N.; HALL, W.; BERNERS-LEE, T. The Semantic Web Revisited. Intelligent Systems, IEEE, v. 21, n. 3, p. 96 –101. doi: 10.1109/MIS.2006.62, 2006.

SOARES, M. DOS S. Metodologias Ágeis Extreme Programming e Scrum para o Desenvolvimento de Software. Revista Eletrônica de Sistemas de Informação
ISSN 1677-3071 doi:10.5329/RESI, v. 3, n. 1. doi: 10.5329/146, 2004.

SPELTA, L. Acessibilidade web: 7 mitos e um equívoco. . Retrieved June 21, 2011, from http://acessodigital.net/art_acessibilidade-web-7-mitos-e-um-equivoco.html, 2009.

STAAB, S.; STUDER, R.; SCHNURR, H.-P.; SURE, Y. Knowledge processes and ontologies. Intelligent Systems, IEEE, v. 16, n. 1, p. 26 – 34. doi: 10.1109/5254.912382, 2001.

STEELS, L. Evolving grounded communication for robots. Trends in Cognitive Sciences, v. 7, n. 7, p. 308–312. doi: 10.1016/S1364-6613(03)00129-3, 2003.

SYCARA, K. P. Multiagent Systems. AI Magazine, v. 19, n. 2, p. 79. doi: 10.1609/aimag.v19i2.1370, 1998.

TANAKA, E. H.; DA ROCHA, H. V. Evaluation of web accessibility tools. Proceedings of the 10th Brazilian Symposium on on Human Factors in Computing Systems and the 5th Latin American Conference on Human-Computer Interaction. Anais... , IHC+CLIHC '11.. p.272–279. Porto Alegre, Brazil, Brazil: Brazilian Computer Society. Retrieved July 24, 2012, from http://dl.acm.org/citation.cfm?id=2254436.2254483, 2011.

THOMAS, D. A.; ELY, R. J. Making Differences Matter: A New Paradigm for Managing Diversity. Harvard Business Review. Retrieved July 24, 2012, from http://hbr.org/1996/09/making-differences-matter-a-new-paradigm-for-managing-diversity/ar/1, 1996.

TREWIN, S. Physical usability and the mobile web. Proceedings of the 2006 international cross-disciplinary workshop on Web accessibility (W4A): Building the mobile web: rediscovering accessibility? Anais... , W4A '06.. p.109–112. New York, NY, USA: ACM. doi: 10.1145/1133219.1133239, 2006.

UNITED NATIONS. Factsheet on Persons with Disabilities. . Retrieved August 6, 2012, from http://www.un.org/disabilities/default.asp?id=18, 2010.

USCHOLD, M.; KING, M. Towards a Methodology for Building Ontologies. . Retrieved October 20, 2011, from http://129.215.202.23/~oplan/documents/1995/95-ont-ijcai95-ont-method.pdf, 1995.

VALLEJO, J. M. B. Exclusão digital dos jovens brasileiros e a importância da escola aberta. InterSciencePlace, v. 0, n. 3. doi: 10.6020/31, 2009.

VALLVERDÚ, J. (ED.). Thinking Machines and the Philosophy of Computer Science. IGI Global. Retrieved July 29, 2012, from http://www.igi-global.com/chapter/thinking-machines-philosophy-computer-science/43695, 2010.

W3C, W. W. W. C. Web Accessibility Initiative (WAI), 2006.

WANG, M.-H.; LEE, C.-S.; HSIEH, K.-L. et al. Ontology-based multi-agents for intelligent healthcare applications. Journal of Ambient Intelligence and Humanized Computing, v. 1, n. 2, p. 111–131. doi: 10.1007/s12652-010-0011-5, 2010.

WATANABE, T.; UMEGAKI, M. Capability survey of user agents with the UAAG 1.0 test suite and its impact on Web accessibility. Universal Access in the Information Society, v. 6, n. 3, p. 221–232. doi: 10.1007/s10209-007-0087-7, 2007.

WONGTHONGTHAM, P.; CHANG, E.; DILLON, T. S. Enhancing Software Engineering Project Information through Software Engineering Ontology Instantiations. Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence. Anais... , WI '06.. p.33–37. Washington, DC, USA: IEEE Computer Society. doi: 10.1109/WI.2006.70, 2006.

WOOLDRIDGE, M.; JENNINGS, N. R. Intelligent agents: theory and practice. The Knowledge Engineering Review, v. 10, n. 02, p. 115–152. doi: 10.1017/S0269888900008122, 1995.

WU, X.; SUN, J. Study on a KQML-Based Intelligent Multi-agent System. 2010 International Conference on Intelligent Computation Technology and Automation (ICICTA). Anais... v. 3, p.466 –469. doi: 10.1109/ICICTA.2010.113, 2010.

ZHANG, T.; LEE, B. Complementary Classification Techniques based Personalized Software Requirements Retrieval with Semantic Ontology and User Feedback. Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on. Anais... p.1358 –1363. doi: 10.1109/CIT.2010.243, 2010.

6 Apêndices

Documento de Requisitos
TohossouWeb
Interface Web do Tohossou

Introdução

Este documento especifica o sistema Torossou, fornecendo aos desenvolvedores as informações necessárias para o projeto e implementação, assim como para a realização dos testes e homologação do sistema.

Visão geral deste documento

Esta introdução fornece as informações necessárias para fazer um bom uso deste documento, explicitando seus objetivos e as convenções que foram adotadas no texto, além de conter uma lista de referências para outros documentos relacionados. As demais seções apresentam a especificação do sistema Tohossou e estão organizadas como descrito abaixo.

- **Seção 1** – Descrição geral do sistema: apresenta uma visão geral do sistema, caracterizando qual é o seu escopo e descrevendo seus usuários.
- **Seção 2** – Requisitos funcionais (casos de uso): especifica todos os requisitos funcionais do sistema, descrevendo os fluxos de eventos, prioridades, atores, entradas e saídas de cada caso de uso a ser implementado.
- **Seção 3** – Requisitos não funcionais: especifica todos os requisitos não funcionais do sistema, divididos em requisitos de usabilidade, confiabilidade, desempenho, segurança, distribuição, adequação a padrões e requisitos de hardware e software.
- **Seção 4** – Descrição da interface com o usuário: apresenta desenhos, figuras ou rascunhos de telas do sistema.

Convenções, termos e abreviações

A correta interpretação deste documento exige o conhecimento de algumas convenções e termos específicos, que são descritos a seguir.

Identificação dos Requisitos

Por convenção, a referência a requisitos é feita através do nome da subseção onde eles estão descritos, seguido do identificador do requisito, de acordo com o esquema abaixo:

[nome da subseção.identificador do requisito]

Por exemplo, o requisito [Recuperação de dados.RF016] está descrito em uma subseção chamada “Recuperação de dados”, em um bloco identificado pelo número [RF016]. Já o requisito não funcional [Confiabilidade.NF008] está descrito na seção de requisitos não funcionais de Confiabilidade, em um bloco identificado por [NF008].

Prioridades dos Requisitos

Para estabelecer a prioridade dos requisitos foram adotadas as denominações “essencial”, “importante” e “desejável”.

- **Essencial** é o requisito sem o qual o sistema não entra em funcionamento. Requisitos essenciais são requisitos imprescindíveis, que têm que ser implementados impreterivelmente.
- **Importante** é o requisito sem o qual o sistema entra em funcionamento, mas de forma não satisfatória. Requisitos importantes devem ser implementados, mas, se não forem, o sistema poderá ser implantado e usado mesmo assim.
- **Desejável** é o requisito que não compromete as funcionalidades básicas do sistema, isto é, o sistema pode funcionar de forma satisfatória sem ele. Requisitos desejáveis são requisitos que podem ser deixados para versões posteriores do sistema, caso não haja tempo hábil para implementá-los na versão que está sendo especificada.

Seção 1
Descrição geral do sistema

Tohossou é uma deidade presente em diversos ramos religiosos africanos que protege as crianças nascidas com déficits físicos ou neurológicos. É também o nome escolhido para o sistema acessado nas nuvens, onde gerentes de projeto poderão cadastrar seus repositórios (SVN) e torna-los "protegidos" pelo sistema.

Sob o olhar do **Tohossou**, todos os documentos passíveis de validação serão avaliados e classificados quanto à acessibilidade. Esta classificação será remetida aos partícipes do projeto para que as providências corretivas sejam feitas. A remessa ocorrerá por e-mail, na periodicidade estipulada pelo gerente do projeto e também estarão disponíveis na própria interface da ferramenta.

Os agentes do **Tohossou**, terão acesso ao repositório e a um conjunto de ontologias que conterão o conhecimento acerca das características de acessibilidade desejadas. Este repositório de ontologias poderá no futuro comportar colaboração de usuários do sistema, mas inicialmente será determinado pelos gestores do serviço.

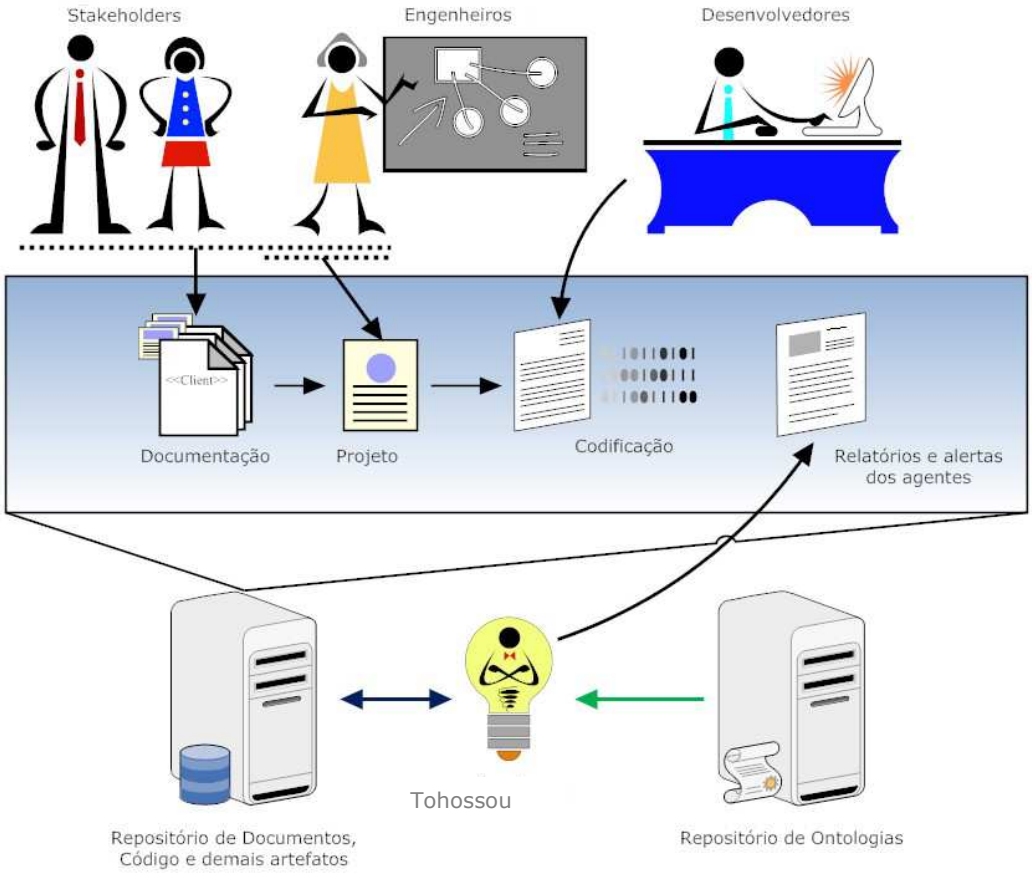


Figura 01 – Diagrama Geral do fluxo de ações do Tohossou

Abrangência e sistemas relacionados

O sistema é um SMA, que funcionará nas nuvens, acessando repositórios independentes na Web, mas não depende dos mesmos para seu funcionamento. Não depende de sistemas externos mas pode possuir agentes funcionando em plataformas de hardware distintas e geograficamente distribuídas.

Descrição dos usuários

Essencialmente os clientes do serviço serão desenvolvedores e gestores de projetos de desenvolvimento de páginas e serviços para a Web que tenham preocupação com aspectos de acessibilidade, mas não desejam submeter suas equipes a uma tarefa adicional de rastrear e validar o código.

Visitante

O visitante natural do serviço, que tem acesso a estatísticas e consultas às características técnicas do serviço. O visitante, uma vez cadastrado torna-se usuário do sistema.

Usuário

Usuário cadastrado mas ainda não vinculado a nenhum projeto. Uma vez cadastrado o usuário pode tornar-se gerente de projeto, ou participante de um projeto existente.

Seção 2

Requisitos funcionais (casos de uso)

Esta seção, apresenta todos os requisitos funcionais, ou casos de uso, do sistema. Em sistemas grandes é comum haver muitos casos de uso e, para facilitar a visualização deste documento, você pode agrupá-los em subseções de casos de uso correlacionados. Os nomes das subseções devem ser únicos e pequenos (3 palavras no máximo) e podem ser formados por palavras, números e/ou abreviações.

Cada um dos casos de uso deve ser descrito em um bloco específico, seguindo o modelo descrito abaixo. O identificador do bloco deve conter o número do caso de uso (por exemplo, [RF001]) e o seu nome. Se os casos de uso forem agrupados em subseções específicas, a numeração deles deve ser reiniciada a cada subseção (dentro de uma mesma subseção, todo caso de uso deve ter um número de identificação único).

Quando a primeira versão deste documento é disponibilizada para a equipe de desenvolvimento, os nomes das subseções e os números dos casos de uso não devem ser modificados ou reaproveitados, para não invalidar referências externas feitas a eles.

Cadastros

O sistema, para poder efetivamente prestar serviços úteis, deve ser alimentado com informações prestadas pelos usuários, além da necessidade explícita de cadastro dos próprios usuários.

[RF001] Cadastro de usuário

Ator: Visitante do site do serviço

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

Interface(s) associada(s): int001 e int002

Entradas e pré-condições: estar na página principal do serviço

Saídas e pós-condições: enviar e-mail de confirmação de cadastro; gravar os dados do usuário; o visitante torna-se usuário (status: Cadastrado)

Fluxo de eventos principal

RF001.1. O usuário clica na interface principal do sistema, no *link* que dá acesso ao cadastro;

RF001.2. O usuário fornece sua senha, a confirmação da senha e e-mail (dados obrigatórios);

RF001.3. O usuário clica em [enviar];

RF001.4. O usuário recebe mensagem de aceite dos dados e de remessa de confirmação por e-mail.

Fluxos secundários (alternativos e de exceção)

RF001.2.a O usuário clica em [cancelar];

RF001.2.1.a o sistema retorna à tela principal [int001];

RF001.4.a Já existe usuário cadastrado com este e-mail;

RF001.4.1.a o sistema retorna à tela de cadastro [int002];

RF001.4.b a senha não confere com a confirmação;

RF001.4.1.b o sistema retorna à tela de cadastro [int002];

[RF002] Cadastro de projeto

Ator: Usuário

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

Interface(s) associada(s): int003 e int004

Entradas e pré-condições: o usuário fez login no sistema

Saídas e pós-condições: O projeto está cadastrado; O usuário torna-se gerente do projeto;

Fluxo de eventos principal

RF002.1. O usuário clica na interface de usuário na função [cadastrar projeto];

RF002.2. O usuário fornece nome do projeto, descrição[opcional], data de início e fim do projeto;

RF002.3. Com a conexão o usuário clica em [cadastrar projeto];

RF002.4. O projeto é listado entre os projetos do usuário.

Fluxos secundários (alternativos e de exceção)

RF002.2.a O usuário clica em [cancelar];

RF002.2.1.a O sistema retorna à tela principal [int004];

RF002.4.b Já existe projeto cadastrado com este nome;

RF002.4.1.b O sistema retorna à tela de cadastro [int003].

[RF003] Convite de usuário

Ator: Usuário - Gerente de projeto

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

Interface(s) associada(s): int004 e int005

Entradas e pré-condições: estar logado e ter projetos sob sua gerência

Saídas e pós-condições: confirmação de envio de e-mail de convite a um futuro usuário; um usuário é criado e marcado como Convidado; este usuário é vinculado a um projeto;

Fluxo de eventos principal

RF003.1. O gerente seleciona projeto;

RF003.2. O gerente clica em [convidar usuário];

RF003.3. O gerente fornece e-mail, *status* que o usuário terá no projeto (gerente, *stakeholder* ou desenvolvedor) e uma mensagem adicional ao usuário;

RF003.4. Uma mensagem de convite é enviada e o gerente recebe confirmação do envio do e-mail;

Fluxos secundários (alternativos e de exceção)

RF003.2.a O usuário clica em [cancelar];

RF003.2.1.a O sistema retorna à tela principal [int004];

RF003.4.b Já existe usuário cadastrado com este e-mail;

RF003.4.1.b O sistema retorna à tela de cadastro [int005].

[RF004] Cadastro de Repositório

Ator: Usuário - Gerente de projeto

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

Interface(s) associada(s): int004 e int006

Entradas e pré-condições: estar logado e ter projetos sob sua gerência

Saídas e pós-condições: Configuração de repositório de um projeto;

Fluxo de eventos principal

RF004.1. O gerente seleciona projeto;

RF004.2. O gerente clica em [incluir repositório];

RF004.3. O gerente fornece a URL, o usuário e a senha do repositório;

RF004.3.1. O sistema testa o repositório;

RF004.4. O gerente confirma a inclusão clicando em [incluir];

RF004.5 O sistema retorna à tela principal [int004].

Fluxos secundários (alternativos e de exceção)

RF004.2.a O usuário clica em [cancelar];

RF004.2.1.a O sistema retorna à tela principal [int004];

RF004.3.1.b O sistema testa a conexão ao repositório e falha;

RF004.3.2.b O sistema retorna à tela de cadastro [int006].

[RF005] Aceite de convite

Ator: Visitante ou Usuário

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

Interface(s) associada(s): int007

Entradas e pré-condições: haver recebido e-mail de convite

Saídas e pós-condições: o visitante torna-se usuário ou usuário confirma a informação;

Fluxo de eventos principal

RF005.1. O convidado clica no link em seu e-mail;

RF005.2. O sistema mostra interface do usuário;

RF005.3. O usuário informa seus dados e senha;

RF005.4. O usuário confirma clicando em [aceitar];

RF005.5 O sistema retorna à tela principal [int004].

Fluxos secundários (alternativos e de exceção)

RF005.2.a O usuário já está cadastrado;

RF005.3.a O usuário confirma ou não a participação;

RF005.4.b O usuário clica em [cancelar];

RF005.5.b O sistema retorna à página principal [int001].

Operações

[RF006] Logar no sistema

Ator: Usuário

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

Interface(s) associada(s): int001 e int004

Entradas e pré-condições: haver se cadastrado

Saídas e pós-condições: o usuário acessa sua interface;

Fluxo de eventos principal

RF006.1. O usuário digita e-mail e senha;

RF006.2. O usuário clica em [entrar];

RF006.3. O sistema autentica o usuário;

RF006.4. O sistema mostra a tela principal [int004].

Fluxos secundários (alternativos e de exceção)

RF006.3.a O e-mail ou senha não conferem;

RF006.4.a O sistema volta à tela principal [int001].

Seção 3

Requisitos não funcionais

Esta seção contém os requisitos não funcionais do sistema. Para uma melhor organização deste documento, utiliza subseções abaixo para agrupar os requisitos não funcionais relacionados.

Os requisitos não funcionais devem ser identificados com um identificador único, da mesma maneira que os requisitos funcionais (casos de uso). Inicia-se a numeração com o identificador NNF001 e prossegue-se incrementando os números à medida que forem surgindo novos requisitos não funcionais.

Usabilidade

Esta subseção descreve os requisitos não funcionais associados à facilidade de uso da interface com o usuário, material de treinamento e documentação do sistema.

[NNF001] O Software deve ser acessível

A ferramenta deve possuir todas as suas interfaces compatíveis com os critérios de acessibilidade do W3C.

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

[NNF002] O Software deve ser compatível com navegadores comuns do mercado

A ferramenta deve ser compatível com os navegadores mais comuns no mercado para não limitar a experiência de uso dos Usuários.

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

Confiabilidade

Esta subseção descreve os requisitos não funcionais associados à freqüência, severidade de falhas do sistema e habilidade de recuperação das mesmas, bem como à corretude do sistema.

[NNF003] Backups Diários

As cópias de segurança devem ser realizadas diariamente, automaticamente e em repositório local e remoto (FTP).

Desempenho

Esta subseção descreve os requisitos não funcionais associados à eficiência, uso de recursos e tempo de resposta do sistema.

[NNF004] Liberação de Memória

Os agentes devem auto-extinguir-se quando totalmente inativos, exceto os essenciais para o uso mínimo do sistema.

Segurança

Esta subseção descreve os requisitos não funcionais associados à integridade, privacidade e autenticidade dos dados do sistema.

[NNF005] Senha criptografada

Deve ser usada técnica de criptografia na gravação da senha dos usuários

Prioridade: ☐ Essencial ☐ Importante ☑ Desejável

[NNF006] Termos de uso e privacidade

Ao cadastrar-se o usuário deve ter acesso à política de privacidade do serviço

Prioridade: ☐ Essencial ☑ Importante ☐ Desejável

Distribuição

Esta subseção descreve os requisitos não funcionais associados à distribuição da versão executável do sistema.

[NNF007] O uso deve ser livre

Deseja-se uso livre da ferramenta, com possibilidade de doações ou custeio de personalizações.

Padrões

Esta subseção descreve os requisitos não funcionais associados a padrões ou normas que devem ser seguidos pelo sistema ou pelo seu processo de desenvolvimento.

[NNF008] RDF

O uso do padrão RDF garantirá a interoperação do SMA com outros sistemas sob o ponto de vista do intercâmbio de ontologias.

[NNF009] XML

O uso do padrão XML garantirá a interoperação do SMA com outros sistemas através de *Web Services*.

Hardware e software

Esta subseção descreve os requisitos não funcionais associados ao hardware e software usados para desenvolver ou para executar o sistema.

[NNF010] O Software de ser constituído de ferramentas de software livre

A ferramenta deve utilizar em todas as suas etapas ferramentas de software livre não somente para minimizar custos como para permitir o livre acesso ao código fonte do sistema.

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

[NNF011] O Software de ser compatível com diversas plataforma de software e hardware

A ferramenta deve ser portável a plataformas de hardware e software comuns no mercado.

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

Seção 4

Descrição da interface com o usuário

[int001] Start Page

Página principal, com estatísticas de uso do serviço, *links* para explicação e documentação de suas funcionalidades. *Link* para *login* e cadastro de novo usuário.

[int002] Página de cadastro de usuário

Convém estudar alguma semelhança com serviços “clássicos” para facilitar o uso. Seria interessante incluir termos de uso do sistema e indicador de força da senha.

[int003] Página de cadastro de Projeto

Nesta página deve-se mostrar o processo de cadastro de projeto.

[int004] Página Principal de usuário

Nesta página deve-se mostrar: os dados do usuário e de seus projetos associados

[int005] Página de convite de usuário

Dentre as propriedades de um projeto, pode-se convidar um usuário fornecendo apenas seu e-mail.

[int006] Página de cadastro de repositório

Dentre as propriedades de um projeto, pode-se convidar um usuário fornecendo apenas seu e-mail.

[int007] Página de aceite de convite

Se o convidado já é usuário ele deve aceitar ou não a participação. Se aceitar, deverá preencher completamente o cadastro de usuário.

Diagrama de Casos de Uso do Usuário

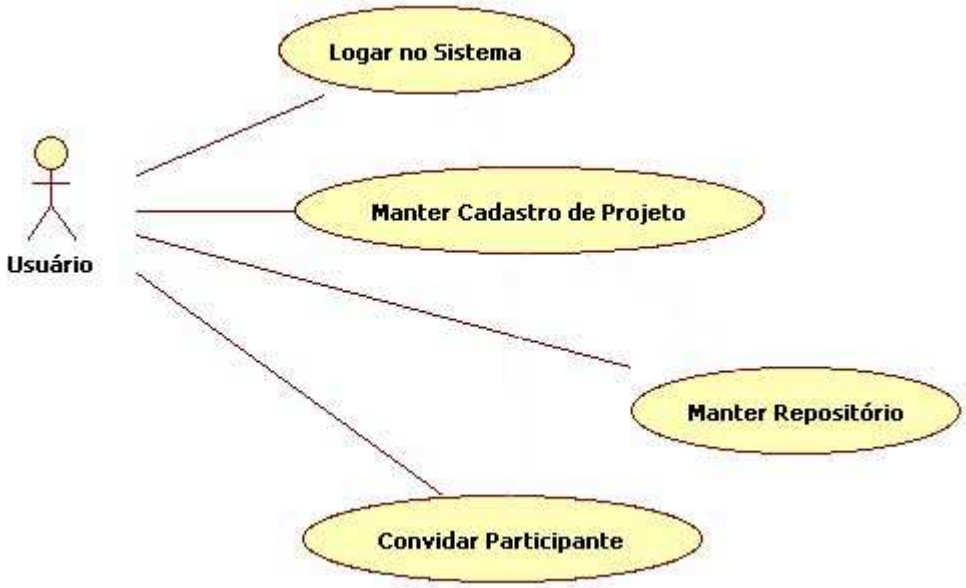


Diagrama de Casos de Uso do Convidado

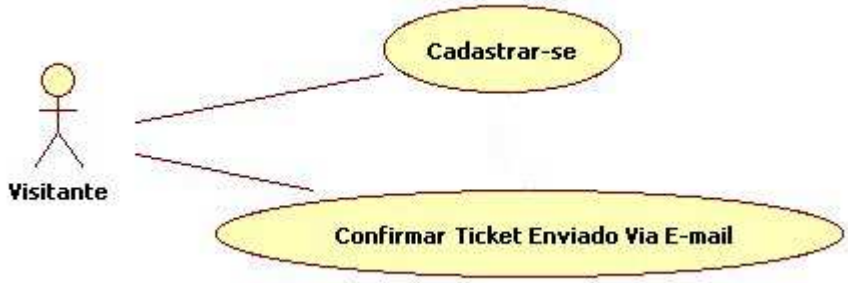


Diagrama de Atividades de Registro de Usuário

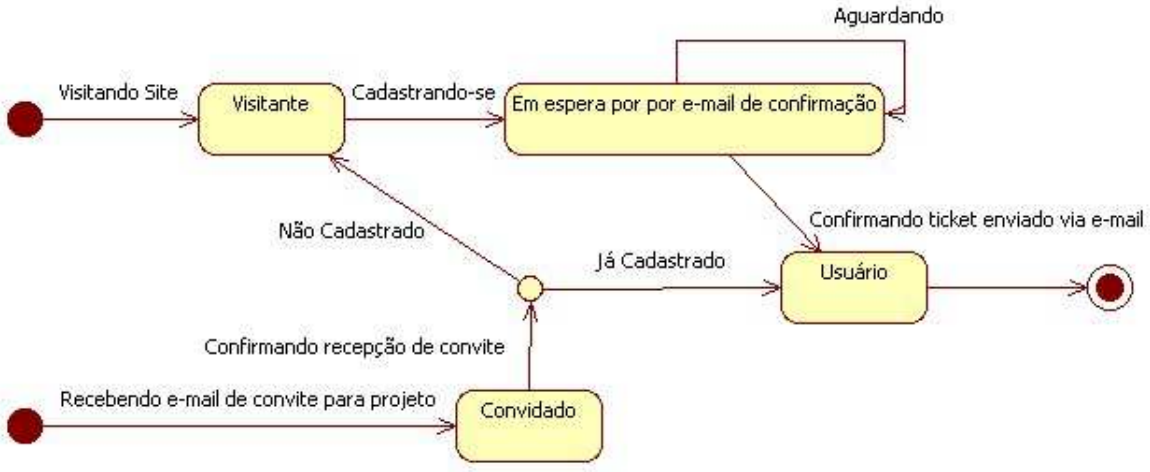


Diagrama de Atividades do Agente Bibliotecário

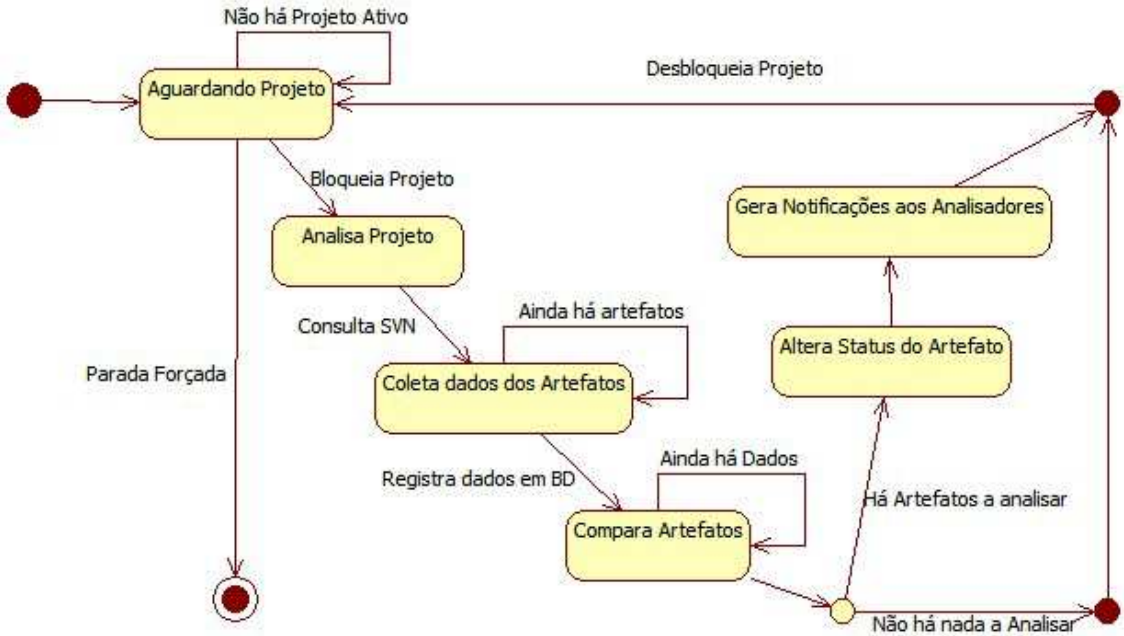


Diagrama de Atividades do Agente Analisador

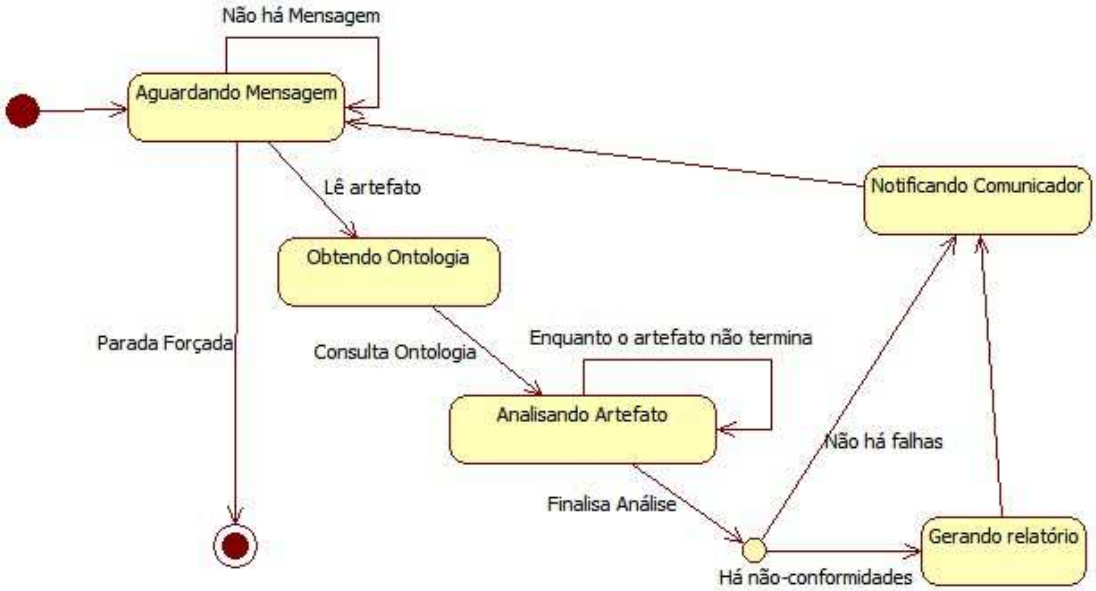
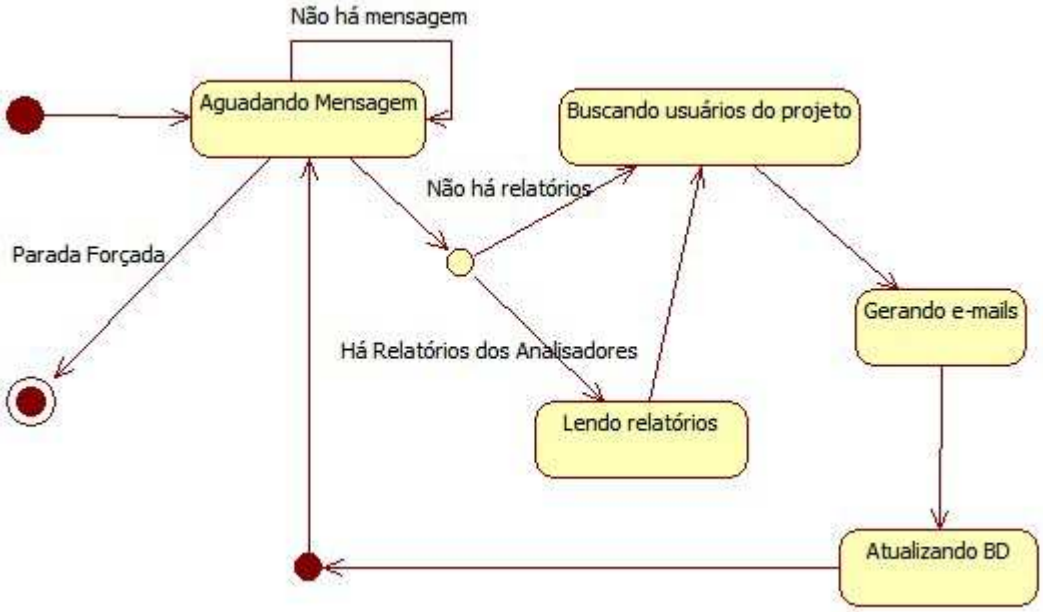


Diagrama de Atividades do Agente de Comunicação



Modelos de Consultas SPARSQL

Retorna todos os objetos da ontologia

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?subject ?object
       WHERE { ?subject rdfs:subClassOf ?object }
```

Retorna todos as classes da sub-classe “Sentences” da ontologia Tohossou_Html_mapping.owl:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX TextConstitution:
<http://www.semanticweb.org/ontologies/2012/3/TextConstitution.owl#>
PREFIX Tohossou_Html_mapping:
<http://www.semanticweb.org/ontologies/2012/3/Tohossou_Html_mapping.owl#>
select ?subject
where {
   ?subject rdfs:subClassOf TextConstitution:Sentences .
}
```

Retorna todos as propriedades da classes “tag” da ontologia Tohossou_Html_mapping.owl:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX TextConstitution:
<http://www.semanticweb.org/ontologies/2012/3/TextConstitution.owl#>
PREFIX Tohossou_Html_mapping:
<http://www.semanticweb.org/ontologies/2012/3/Tohossou_Html_mapping.owl#>

SELECT DISTINCT ?property
WHERE {
  ?s a Tohossou_Html_mapping:tag;
     ?property ?o .
}
```

Retorna todas as propriedades dos Indivíduos da classe “tag” da ontologia Tohossou_Html_mapping.owl:

```
Recuperação de dados de propriedades de objetos (individuals)
(NA HORA DE FAZER TEM QUE FAZER UMA PROPRIEDADES POR VEZ (VERIFICAR
INCOMPATIBILIDADE DE PROPRIEDADES))

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX tc:
<http://www.semanticweb.org/ontologies/2012/3/TextConstitution.owl#>
PREFIX thm:
<http://www.semanticweb.org/ontologies/2012/3/Tohossou_Html_mapping.owl#>

SELECT ?ind ?banned ?requires ?wants ?Required ?finishedBy ?StartedBy
WHERE {
  ?ind rdf:type thm:tag .
    ?ind tc:finishedBy ?finishedBy .
?ind tc:StartedBy ?StartedBy .
?ind thm:wants ?wants .
?ind thm:Required ?Required .
?ind thm:requires ?requires .
?ind thm:banned ?banned .

}
```