UNIVERSIDADE DO VALE DO RIO DOS SINOS
CIÊNCIAS EXATAS E TECNOLÓGICAS
PROGRAMA INTERDISCIPLINAR DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

RODRIGO MACHADO HAHN

Sim: Uma Arquitetura para Determinação de Similaridade entre Trilhas

Proposta de dissertação submetida à avaliação
Como requisito parcial para obtenção de grau de
Mestre em Computação Aplicada

Prof. Dr. Jorge Luis Victória Barbosa
Orientador

São Leopoldo, Maio de 2011

SUMÁRIO

1 INTRODUÇÃO 7
1.1 Motivação 7
1.2 Definição do Problema 9
1.3 Objetivos e Contribuição 10
1.4 Organização do Trabalho 10
2 CONCEITUAÇÃO 12
2.1 Computação Ubíqua 12
2.2 Trilhas 13
2.3 Ontologias 15
2.3.1 Definição e Conceituação 15
2.3.2 Elementos Componentes das Ontologias 16
2.3.3 Representação 17
2.3.4 Ferramentas 19
2.4 Similaridade 20
2.4.1 Similaridade Léxica 20
2.4.2 Similaridade Semântica 22
2.5 Considerações Sobre o Capítulo 22
3 TRABALHOS RELACIONADOS 23
3.1 Computing Semantic Similarity Using Ontologies 23
3.2 Measuring Similarity Between Ontologies 28
3.3 *TrailTRECer* 31
3.4 Avaliação dos Trabalhos Pesquisados 32
3.5 Considerações sobre o Capítulo 34
4 SIM 35
4.1 Interface de Acesso 36
4.2 Módulo Analisador de Trilhas 39
4.3 Algoritmos de similaridade 40
4.3.1 Comparação de Eventos 40
4.3.2 Comparação de faixas de tempo 41
4.3.3 Sequências de Visitação (locais) 41
4.4 Módulo de Recuperação de Resultados 43

5 ASPECTOS DE IMPLEMENTAÇÃO E AVALIAÇÃO 45
5.1 LOCAL 47
5.2 UBITRAIL 50
5.3 Implementação 54
5.4 Avaliação 58
6 CONSIDERAÇÕES FINAIS 62
6.1 Contribuições 62
6.2 Aplicações futuras 62
6.3 Trabalhos Futuros 63
7 REFERÊNCIAS 65

Lista de Figuras

Figura 1. Taxonomia da computação ubíqua (Adaptado de [8]) 13
Figura 2. Exemplo de composição de trilhas 14
Figura 3. Semantic MediaWiki , ... 18
Figura 4. Relação entre os diferentes tipos de OWL 18
Figura 5. Representação em OWL DL de algumas informações sobre o Mobilab 19
Figura 6. Captura de tela no Protégé OWL (usando o OWLviz) 19
Figura 7. Processo de *spreading* baseado em conjuntos (conforme [18]) 24
Figura 8. Processo de *spreading* baseado em grafos (conforme [18]) 25
Figura 9. Análise comparativa entre dois perfis de usuário (conforme [18]) 27
Figura 10. Análise de monotonicidade entre dois perfis de usuário (conforme [18]) .. 27
Figura 11. Duas ontologias de exemplo (conforme [19]) 30
Figura 12. Arquitetura do TrailTRECer 31
Figura 13. Visão geral da arquitetura 35
Figura 14. Exemplo de composição de trilhas 39
Figura 15. Módulo Analisador de Trilhas operando em modo automático 40
Figura 16. Algoritmo *Graph spreading* aplicado a termos presentes nas trilhas 41
Figura 17. Determinação do coeficiente de similaridade de tempo 41
Figura 18. Algoritmo Tree Edit Distance aplicada às trilhas 42
Figura 19. Diferenças no segundo nível de composição da trilha da Figura 18 42
Figura 20. Visão Geral da integração entre o SIM, LOCAL e Ubitrail 46
Figura 21. Arquitetura do LOCAL 47
Figura 22. Sistema de perfis do LOCAL 48
Figura 23. Objetos de aprendizagem 49
Figura 24. Sistema de comunicação 49
Figura 25. Funcionamento do tutor 50
Figura 26. Modelo geral do Ubitrail (conforme [21]) 51
Figura 27. Composição de trilhas no Ubitrail 51
Figura 28. Detalhamento da camada TrailServices do UbiTrail 52
Figura 29. Mapeamento dos dados de tracking do LOCAL para as trilhas 55
Figura 30. Criação de ontologia usando a biblioteca SemWeb 56
Figura 31. Ontologia resultante do código fonte mostrado na Figura 30 57
Figura 32. Representação gráfica da ontologia criada 58

Figura 33. Cenário do teste simulado 58

Lista de Tabelas

Tabela 1. Valores típico para correspondência de strings 30
Tabela 2. Comparativo entre abordagens de determinação de similaridade 33
Tabela 3. Formato básico assumido para as trilhas 36
Tabela 4. Serviços oferecidos pela interface de consulta 37
Tabela 5. Tipos de comparação suportados 38
Tabela 6. Estratégias de *cache* suportadas 43
Tabela 7. Descrição dos serviços especializados no Ubitrail (conforme [21]) 53
Tabela 8 - MD5 *versus* CRC32 para a criação de *hashes* **Erro! Indicador não definido.**

1 INTRODUÇÃO

Atualmente, os estudos sobre mobilidade em sistemas distribuídos vêm sendo impulsionados pela proliferação de dispositivos eletrônicos portáteis (por exemplo, celulares, *handhelds*, *tablet* PCs e *notebooks*) e pela exploração de novas tecnologias de interconexão baseadas em comunicação sem fio (tais como, WiFi, *Bluetooth*, 3G e WiMAX). Este novo paradigma distribuído e móvel é denominado Computação Móvel [1,2].

Os últimos anos têm sido uma época de grandes transformações no que diz respeito às tecnologias de comunicação. Devido à popularização da computação móvel (dispositivos menores, mais baratos, com maior autonomia) e das redes sem fio, o mundo tem estado cada vez mais conectado. Com o advento das Tecnologias de Localização [3,4], surgiram diversos tipos de aplicações. Essas aplicações utilizam a infraestrutura de comunicação disposta no ambiente para oferecer serviços personalizados. A utilização dessas tecnologias possibilita que se tenha um histórico completo dos deslocamentos/interações de uma entidade qualquer em um ambiente. Esse histórico de deslocamentos chama-se trilha[5,6].

Através da análise dos dados presentes em uma trilha, pode-se obter informações a respeito de uma entidade ou grupo de entidades. Essa análise possibilita o desenvolvimento de diferentes tipos de aplicações. Por exemplo, pode-se usar o histórico de informações de deslocamento de um veículo em movimento para inferir, com determinado grau de confiabilidade, seu próximo destino. Deste modo, o objetivo deste trabalho, nesse contexto, é propor, implementar e validar uma arquitetura de detecção de similaridades entre trilhas de entidades genéricas. Essas trilhas serão obtidas externamente – o escopo do trabalho limita-se a fornecer os meios através dos quais essa detecção de similaridade ocorrerá, bem como os resultados. Também será realizada a integração da solução desenvolvida com uma arquitetura de suporte à educação ubíqua, a fim de validar a proposta.

As seções seguintes descrevem o contexto no qual o trabalho está inserido e detalham os objetivos perseguidos na realização do mesmo. A organização geral do texto será apresentada ao final desta seção.

1.1 Motivação

Localização, como pode ser visto em [3] e [4], é um importante tópico relacionado à computação móvel e ubíqua [7,8]. Essa frente de pesquisa propõe a determinação da localização física de um dispositivo móvel (por exemplo, uma sala em um prédio ou um endereço específico em uma rua). A informação pode ser obtida através de posicionamento de

satélites (GPS, A-GPS) e/ou através de antenas *wireless* (CDMA, GSM e Wifi), e a precisão atual da localização permite a implementação de aplicações comerciais. Além disso, a rápida proliferação de antenas *wireless* torna previsível uma crescente precisão da localização, estimulando a criação de serviços baseados em localização (LBS, *location based services*) [9,10]. A eficácia da localização baseada em satélites é bastante reduzida em ambientes urbanos (com prédios) e locais fechados (*indoor*). O uso da localização seria consideravelmente ampliado se pudesse ser contínuo, em qualquer momento e em qualquer. Consequentemente, estão surgindo alternativas ao uso de satélites, tais como, a integração de GPS com telefonia móvel (A-GPS [11,12]) e o posicionamento baseado em antenas *wireless* (telefonia celular [13], *wifi* [14]ou em ambas[4]). Baseando-se nesses estudos, pode-se prever um cenário de médio prazo, onde a sociedade será permeada de dispositivos móveis sempre conectados a uma rede de comunicação e com informação precisa da sua localização. Neste cenário, a computação ubíqua será estimulada, visto que a mesma estará disponível em qualquer tempo e lugar.

Nos últimos anos, o uso conjunto de contextos e perfis de usuários vem sendo considerado uma oportunidade para a distribuição de conteúdo. Além disso, o aprimoramento e a ampla adoção dos sistemas de localização vêm estimulando ainda mais o acompanhamento da mobilidade, viabilizando o uso de trilhas. Considera-se que a Ciência de Trilhas [15] é uma evolução da proposta de uso conjunto de contextos e perfis.

Uma trilha pode ser formada por diversos tipos de dados. Estes dados compreendem vários níveis de informação, como por exemplo: 1) propriedades que descrevem uma determinada entidade; 2) dados contextuais e/ou geográficos; 3) eventos ocorridos em um determinado local, e; 4) dados temporais (início, fim e duração de um determinado evento).

O ponto forte da organização de informações em trilhas é justamente observar e inferir sobre este rico conjunto de dados interligados. A organização de informações em trilhas se presta à detecção de conjuntos, relacionamentos e tendências entre as entidades cujas trilhas estão devidamente catalogadas. Isso pode ser feito através da detecção de elementos similares dentro das trilhas.

A determinação da similaridade entre dois conteúdos quaisquer é um problema importante em diversas áreas de pesquisa, como a mineração de dados, a publicidade na mídia e o alinhamento de grupos especialistas. Esta determinação de similaridade pode ocorrer de duas formas distintas: primeiramente, através da comparação de termos em relação ao vocabulário, incluindo palavras e expressões (similaridade léxica [16]); também é possível

buscar similaridades nos conceitos em si, ou seja, o significado das palavras e sentenças (similaridade semântica [17]).

Pode-se pensar em uma ampla gama de aplicações para um sistema que possibilite a determinação de similaridade entre trilhas. Por exemplo: a) encontrar similaridades entre duas trilhas de objetos; b) encontrar veículos fora de rota; c) detectar engarrafamentos; d) prever passos futuros em uma trilha com determinado grau de certeza; dentre muitas outras. De modo a tornar esta funcionalidade acessível às mais diversas aplicações, imagina-se que esta seja oferecida como um serviço/API pelo ambiente, seja este um contexto lógico ou ambiente computacional. Isso seria vantajoso no sentido de oferecer as mesmas técnicas de determinação de similaridade para todas as aplicações, de maneira unificada.

Através de pesquisa realizada constatou-se que não existe ainda um modelo robusto para a determinação de similaridade entre trilhas de entidades genéricas, acessível a aplicações externas. Os trabalhos atuais ([18,19,20]) preocupam-se mais em estabelecer o armazenamento de informações relevantes, organizar essas informações em repositórios, e realizar inferências sobre estes conjuntos de dados, ou então na organização da arquitetura em si, e não na questão da similaridade e de prover acesso a essas funcionalidades.

1.2 Definição do Problema

Quando se imagina a quantidade de informações que uma trilha pode conter, rapidamente percebe-se o potencial para uma ampla gama de aplicações (como data mining, matching, recrutamento & seleção, recomendação, determinação e otimização de trajetos, etc...). Dentre todas estas, percebe-se um elemento comum que as torna possíveis: a determinação de similaridade entre dois conjuntos de informações. Através da similaridade, torna-se possível perceber a formação de padrões e a identificação de tendências dentro de conjuntos de dados. No entanto, através da pesquisa realizada pôde-se perceber que ainda não existe uma arquitetura que possibilite responder essas questões de maneira parametrizável e aberta a aplicações. A parametrização é importante, visto que, dependendo da aplicação, pode ser interessante comparar apenas certa dimensão em um conjunto de trilhas (por exemplo, apenas localização geográfica e informações temporais, no caso de uma aplicação de otimização dinâmica de rotas e/ou transporte). Além disso, a disponibilidade dessas funções como serviço faz todo sentido, uma vez que diversas aplicações podem querer acessar simultaneamente uma trilha que descreve uma determinada entidade.

Sendo assim, este trabalho busca responder às seguintes questões: **é possível criar uma arquitetura com um padrão aberto, que ofereça como serviço a capacidade de**

detectar graus de similaridade entre as diferentes dimensões de dados existentes nas trilhas de um conjunto de entidades? Além disso, caso seja possível, como seria estruturada tal arquitetura? E quais funcionalidades seriam oferecidas?

1.3 Objetivos e Contribuição

Este trabalho tem como objetivo definir, desenvolver e avaliar uma arquitetura para a determinação da similaridade entre trilhas de entidades. Mais especificamente, o objetivo do trabalho é possibilitar a realização de comparações entre instâncias de trilhas que representam entidades. A definição de entidade é a mesma empregada pelo UbiTrail [21], onde uma entidade é dita como "*(...)uma pessoa acessando recursos computacionais (por exemplo, um smartphone) ou um objeto móvel (por exemplo, um veículo).*". Desse modo, é possível representar quaisquer elementos necessários, para qualquer aplicação específica.

O SIM irá disponibilizar serviços pelos quais será possível obter uma medida que indique a similaridade entre trilhas. Esses serviços serão parametrizáveis, isto é, eventuais aplicações poderão escolher qual serviço desejam acessar. Os processos utilizados para a detecção da similaridade serão determinados com base em informações presentes na bibliografia existente.

Como objetivos específicos deste trabalho, temos:

1) Propor, especificar e modelar uma arquitetura capaz de determinar o grau de similaridade de trilhas que descrevam entidades genéricas;
2) Implementar uma instância da arquitetura, aplicando os conceitos de Engenharia de Software e Padrões de Desenvolvimento de Software;
3) Validar a arquitetura, através de testes focados em aplicações de exemplo, que utilizem as funcionalidades providas pela instância implementada.

A arquitetura disponibilizará a estrutura que gera as comparações, os meios para invocá-la em aplicações, e uma maneira de armazenar resultados de comparações já realizadas, bem como uma maneira de recuperá-los rapidamente, no caso de terem sido previamente realizadas. Com isso, espera-se oferecer uma solução robusta, que possibilite responder as questões levantadas no Item 1.2.

1.4 Organização do Trabalho

O texto está organizado em 6 capítulos. **O Capítulo 2** descreve conceitos necessários para o entendimento desse trabalho. Em seguida, o **Capítulo 3** apresenta trabalhos relacionados à arquitetura proposto. O **Capítulo 4** apresenta a arquitetura SIM.

O **Capítulo 5** apresenta aspectos relacionados ao desenvolvimento do protótipo do SIM, bem como a avaliação da arquitetura, através da integração desta com outros dois ambientes relacionados a uma aplicação específica na área de educação ubíqua. O **Capítulo 6** apresenta as considerações finais, bem como algumas idéias de aplicações que possam vir a ser criadas no contexto da arquitetura proposta.

2 CONCEITUAÇÃO

Este capítulo visa elucidar conceitos importantes para a compreensão deste trabalho. Os tópicos analisados serão os seguintes: a) a computação ubíqua e suas implicações na coleta de informações; b) as trilhas propriamente ditas, apresentando a computação ciente de trilhas como uma evolução da computação ciente de contexto; c) ontologias e suas aplicações no contexto deste trabalho, e; d) os conceitos relacionados a similaridade, incluindo a determinação de similaridade entre conjuntos de dados. A revisão sobre ontologias é importante para os métodos de determinação de similaridade empregados na arquitetura; além disso, o texto incorpora uma revisão feita sobre o estado da arte na área de ontologias, ferramentas e técnicas de representação, realizada em 2009. A **Seção 2.4** apresenta brevemente algumas abordagens sobre determinação de similaridade léxica e semântica, essas sendo expostas em maiores detalhes no **Capítulo 4**, dedicado aos trabalhos relacionados.

2.1 Computação Ubíqua

A Computação Móvel [1,2] trouxe a possibilidade de utilizar dispositivos computacionais fora de um ambiente fixo, ou seja, ao utilizar um dispositivo móvel, o usuário leva seu ambiente de trabalho consigo. Com o passar do tempo, esses dispositivos móveis foram se tornando cada vez menores e mais populares. Além disso, foi possível embarcar neles uma capacidade computacional significativa. Mark Weiser [7], por exemplo, preveu um futuro onde objetos triviais seriam dotados de capacidade computacional. Com dispositivos de tamanho reduzido e poder computacionais razoável embutidos no ambiente, tornou-se possível disponibilizar serviços de suporte a qualquer hora, em qualquer lugar ou ocasião. A computação, com o passar do tempo, deixa de pertencer exclusivamente ao que hoje chama-se computadores, e move-se para o ambiente. Nesse novo paradigma os dispositivos computacionais estão embarcados no ambiente, de maneira transparente para o usuário. Esses dispositivos podem obter informações a partir do ambiente, e usá-las para configurar serviços oferecidos aos usuários. Em outras palavras, os espaços/ambientes tornam-se “inteligentes”.

Ao aliar os recursos de mobilidade juntamente com a capacidade de oferecer serviços através do ambiente, surge a Computação Ubíqua ([8]). Um dispositivo móvel pode, enquanto se desloca, consultar informações sobre o ambiente, como serviços oferecidos por dispositivos embarcados (talvez invisíveis) e adaptar-se com base nessas informações. Isso pode envolver configuração automática de redes e dispositivos, adaptação de conteúdo, entre outras funções.

A computação ubíqua, neste caso, pode ser considerada uma como uma extensão da computação móvel (conforme pode ser visto na **Figura 1**).

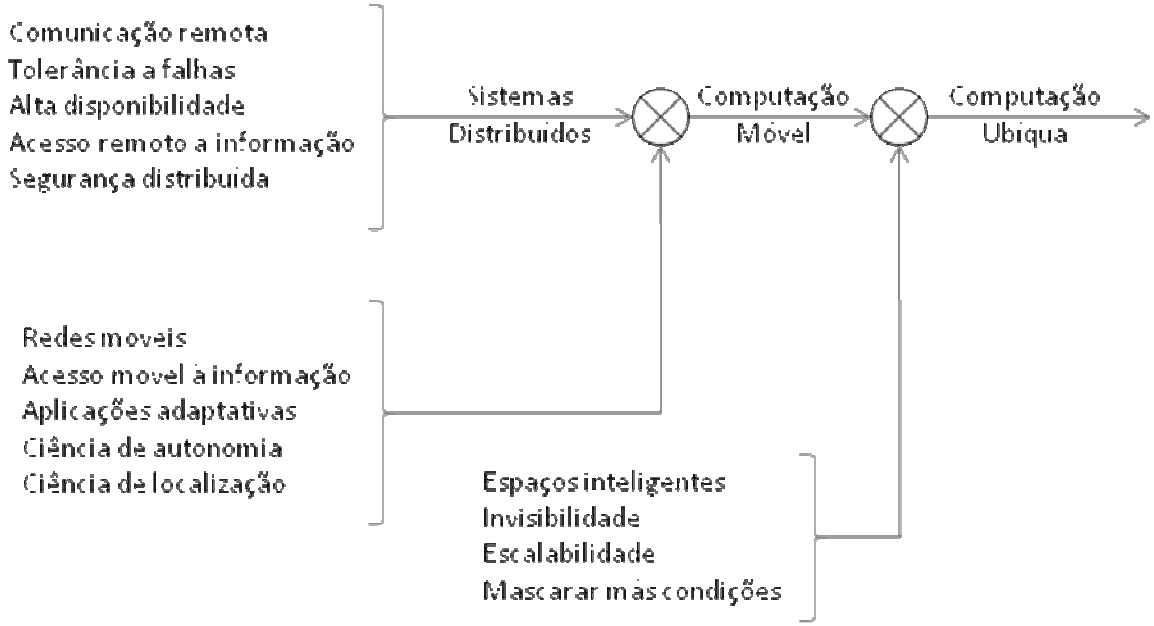

Figura 1. Taxonomia da computação ubíqua (Adaptado de [8])

Localização, como pode ser visto em [4] é um importante tópico relacionado à computação ubíqua. Essa frente de pesquisa propõe a determinação da localização física dos dispositivos móveis dos usuários. Essa informação de localização pode ser obtida através de diversos métodos, como posicionamento de satélites (GPS, A-GPS) e antenas sem fio (3G, GPS, WiFi). A precisão que pode ser obtida hoje em dia já é suficiente para a implementação de aplicações comerciais. Além disso, com a popularização dos dispositivos móveis, essa precisão tende a crescer. Baseando-se nisso, pode-se imaginar um cenário futuro repleto de dispositivos móveis sempre conectados e com dados de localização sempre disponíveis. Diversas aplicações surgem neste cenário, como aplicações voltadas para a área de educação [22,23] e comércio ubíquo [24].

2.2 Trilhas

Nos ambientes de computação móvel, o acompanhamento da mobilidade permite a adaptação das aplicações aos contextos físicos percorridos pelo usuário. Nos últimos anos, o uso conjunto de contextos e perfis de usuários vem sendo considerado uma oportunidade para a distribuição de conteúdo. Além disso, o aprimoramento e a ampla adoção dos sistemas de localização vêm estimulando ainda mais o acompanhamento da mobilidade, viabilizando o uso de trilhas. Considera-se que a Ciência de Trilhas é uma evolução da proposta de uso conjunto de contextos e [21].

Nos últimos anos a evolução dos dispositivos móveis e das redes sem fio de alta velocidade vem impulsionando a adoção da Computação Móvel [1,2]. Além disso, o aprimoramento e a proliferação de Sistemas de Localização[3,4] vêm estimulando a adoção de soluções que considerem com precisão a localização dos usuários na prestação de serviços. Serviços mais aprimorados consideram as informações de contexto, tornando a computação ciente de contexto. O uso conjunto de perfis de usuários e informações do ambiente vem sendo considerado uma oportunidade para a distribuição de conteúdo contextualizado. Recentemente, estudos mostraram que o acompanhamento do usuário em sistemas de computação móvel com suporte à localização, pode ser usado para o registro do histórico dos contextos físicos visitados durante um período de tempo. Esse registro normalmente recebe a denominação de Trilha [5,6]. As trilhas registram atividades de um usuário nos contextos percorridos, mantendo assim, um histórico de seus deslocamentos e de sua atuação em cada contexto (**Figura 2**). Nessa proposta, considera-se que a Ciência de Trilhas constitui uma evolução da proposta de uso conjunto de contextos e perfis de usuários. As trilhas contêm um registro histórico das atividades realizadas em todos os contextos visitados em um período de tempo e, portanto, podem ser usadas para a determinação precisa do perfil das entidades, possibilitando uma série de aplicações específicas [21,25].

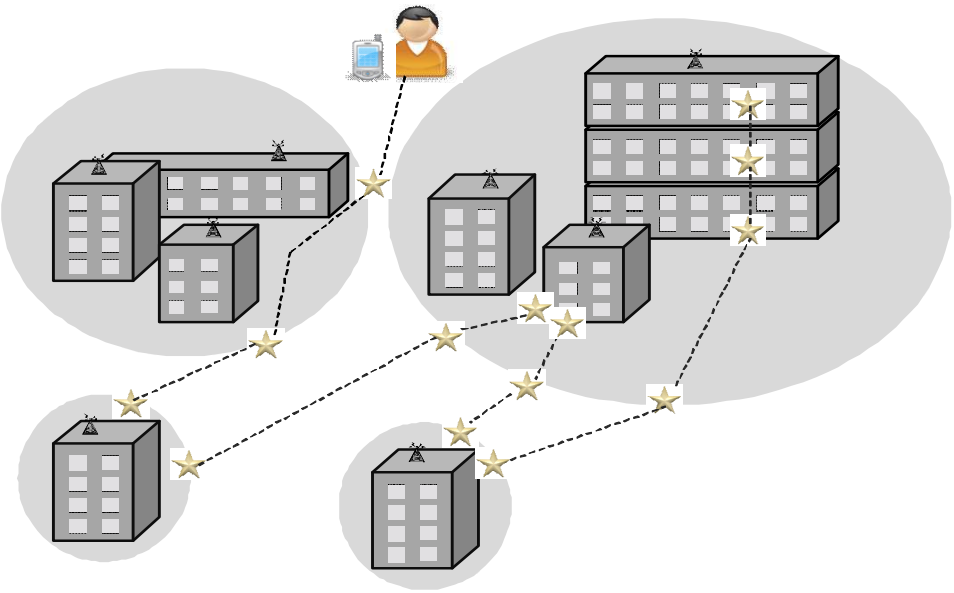

Figura 2. Exemplo de composição de trilhas

2.3 Ontologias

2.3.1 Definição e Conceituação

O termo *Ontologia*[1] é bastante antigo, tendo suas raízes originalmente na área da Filosofia. Especificamente em Ciência da Computação, ontologia é uma maneira de representar o conhecimento. Essa representação geralmente toma a forma de definições conceituais, e de relacionamentos entre estas definições.

Assim como a noção de *agente*, o termo *ontologia* ainda hoje provoca discussões. Uma das definições mais aceitas foi proposta em 1992, por Tom Gruber [26,27]. O autor postulou que uma ontologia pode ser definida como a "*especificação de uma conceitualização*". Isso quer dizer que uma ontologia é a descrição (formal ou informal) dos conceitos – e relacionamentos entre estes conceitos – que podem existir para uma determinada comunidade de agentes. Sendo assim, pode ser encarada como um "contrato" para troca de informações entre estes agentes. Neste âmbito, um agente se compromete com a ontologia quando suas ações são consistentes com as definições ontológicas às quais este se propõe a cumprir.

Geralmente a representação de conhecimento através de ontologias trata de um domínio específico, fornecendo uma visão simplificada do mundo, de acordo com a conceituação com a qual a ontologia está comprometida.

Em termos de implementação, as ontologias são descritas através de uma série de *"vocabulários específicos, usados para descrever certas realidades"* [28]. Isso quer dizer que elas trazem declarações que descrevem de forma explícita o significado dos termos do vocabulário. Dessa forma, ontologias constituem modelos formais, que podem ser posteriormente manipulados por computadores. Assim, ontologias possibilitam que o conhecimento seja compartilhado entre os agentes membros de uma sociedade, e reutilizado tanto por agentes computacionais quanto por seres humanos.

As ontologias se dividem em duas categorias: de domínio e de aplicação. Ontologias de domínio tratam de conceitos específicos a um domínio de conhecimento, por exemplo, descrição de perfis de usuários e de objetos de aprendizagem. Por sua vez, ontologias de aplicação tratam de soluções para problemas específicos dentro de um domínio, como por

---

[1] Do grego *οντοσ λογοι*, significando *conhecimento de/sobre o ser*. *Οντοσ* ("*ontos*") é o infinitivo em grego do verbo *ser*.

exemplo, determinar os melhores objetos de aprendizagem a encaminhar para determinados usuários, de posse dessas duas ontologias de domínio.

2.3.2 Elementos Componentes das Ontologias

Os elementos que podem compor uma ontologia são muitos. No entanto, grande parte das ontologias formais atuais apresenta similaridades na maneira como são descritas.

Em sua complexidade mínima, ontologias são descritas como tendo um vocabulário que descreve um modelo de domínio específico e propriedades e relações entre estes itens de domínio. Os principais elementos que podem ser utilizados para definir ontologias são:

a) Classes - são os conceitos descritos pela ontologia. Uma classe pode ser descrita como um objeto ao qual é vinculado um conjunto de propriedades que o caracterizam. Classes podem ser dispostas de maneira hierárquica, onde as classes filhas herdam as propriedades da classe pai (tudo aquilo que é verdadeiro para a classe pai também o é para a classe filha). Além disso, classes podem ser classificadas quanto a seus membros como intensionais ou extensionais. Uma classe (ou conjunto) é dita extensional se os seus membros são explicitamente enumerados, enquanto que uma classe é intensional se esta "surge" quando se aplica algum tipo de regra. Por exemplo, a classe "professor" que trabalha no PIPCA na UNISINOS é extensional. Já "pessoa que acorda cedo e toma café" é intensional, porque não existe uma lista específica dessas pessoas; o conjunto é inferido com base em suas propriedades;
b) Indivíduos - uma instância individual de uma classe é chamada de indivíduo. A princípio, os indivíduos não são parte componente das ontologias, mas as ontologias são utilizadas para representar e classificar os indivíduos;
c) Atributos - em uma ontologia, as instâncias das classes são descritas com base em características independentes, relacionadas especificamente aos objetos aos quais descrevem. Essas características são denominadas atributos. Os atributos de uma classe podem ser expressos na forma (*sujeito, predicado, valor*). Por exemplo, a mão de uma pessoa possui dedos. Isso quer dizer que a classe "Mão" implementa um atributo chamado "possui", com o valor "dedos". Isso pode ser expresso da seguinte forma: ("Mão", "possui", "Dedos"). Os atributos podem possuir uma cardinalidade associada. Por exemplo, uma pessoa normalmente possui cinco dedos em cada mão. Nesse caso, a classe Mão implementa o atributo "possui", com o valor "dedos", tendo como cardinalidade máxima e mínima 5;

d) Relacionamentos - Em uma ontologia, os relacionamentos definem como uma determinada instância de uma classe se relaciona com as demais. Pode-se dizer que eles descrevem a semântica do domínio específico sobre o qual a ontologia trata. Os relacionamentos geralmente possuem um tipo de dados associado, que especifica a natureza do relacionamento entre uma classe e outra. De um modo geral, é dito que quanto mais relacionamentos a ontologia suporta, maior é a sua expressividade. Os relacionamentos possibilitam descrever as classes da ontologia em termos de uma estrutura hierárquica (*taxonomia*). Nem toda taxonomia é uma ontologia: para ser uma ontologia, é necessário que as classes se relacionem com outras classes.

e) Axiomas - Axiomas são declarações lógicas relacionadas à interpretação dos relacionamentos e conceitos dispostos em uma ontologia. Por exemplo, para uma ontologia que descreva genericamente as relações familiares:

```
∀ A,B
((Pessoa (A) ^ possui_irmao (B)) →
Pessoa (B) ^ possui_irmao (A)))
```

Isso significa que pode-se dizer que para toda Pessoa A que <possui_irmao> B, este também <possui_irmao>, que é o próprio A. Os axiomas são empregados na descrição de ontologias formais. Uma ontologia é dita formal (ou pesada) se esta emprega restrições nos relacionamentos na forma de axiomas.

2.3.3 Representação

A OWL (*Web Onthology Language*) [29] representa uma família de linguagens com o objetivo de representar conhecimento. É a linguagem de ontologias “oficial” do W3C, constituindo uma das tecnologias fundamentais por trás da Web Semântica. Esta iniciativa visa embutir significado nas informações disponíveis na internet, tornando a informação legível por máquinas e facilitando o desenvolvimento de agentes que trabalhem com estas informações. Este princípio é o mesmo da iniciativa abordada na **Figura 3**, que é um exemplo de iniciativa de descrição geográfica através do uso de de ontologias.

A OWL foi baseada em esforços anteriores de representação do conhecimento, na forma das linguagens OIL [30] (*Ontology Inference Layer*) e DAML+OIL [31] (*Darpa Agent Markup Language*). Atualmente, é comum representar as ontologias OWL na sintaxe do padrão RDF (Resource Description Framework) [36], que permite expressá-las fluentemente e serializá-las em XML.

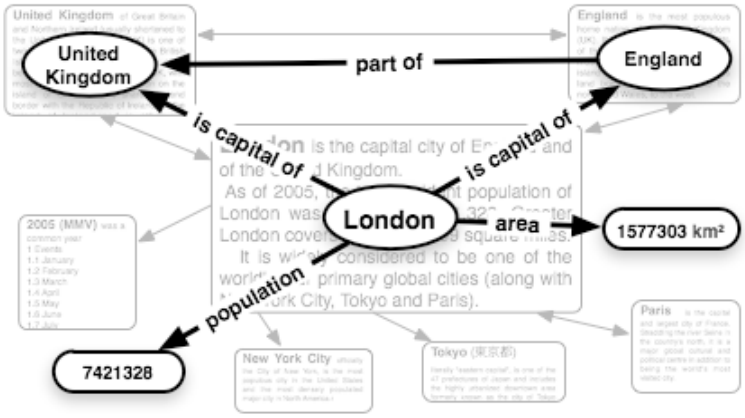

Figura 3. Semantic MediaWiki.

Existem três tipos de OWL, cada qual provendo níveis diferentes de expressividade e de sofisticação na representação do conhecimento. São eles:

a) **OWL Lite** – possibilita a criação de ontologias com restrições simples, representando-as de forma hierárquica. É a menos complexa das três formas existentes da OWL;

b) **OWL Full** – provê uma maior expressividade do que a OWL Lite, porém não garante a computabilidade, isto é, as conclusões obtidas a partir da ontologia não serão sempre computáveis. Isso torna difícil o suporte desta forma da OWL por parte dos computadores, por exemplo, mecanismos de inferência;

c) **OWL DL** – possui maior expressividade do que a OWL Lite, porém garante a computabilidade, representando um meio termo entre a OWL Lite e a OWL Full.

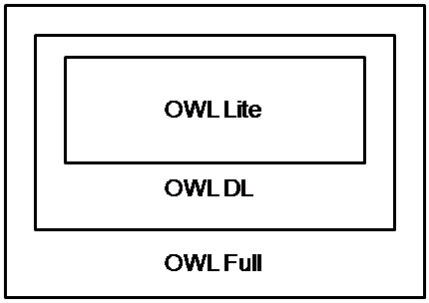

Figura 4. Relação entre os diferentes tipos de OWL

As três formas da OWL estão organizadas de acordo com sua capacidade de representatividade e computabilidade segundo a **Figura 4**, isto é, toda ontologia OWL Lite válida é também uma ontologia OWL DL válida; por sua vez, toda ontologia OWL DL válida é também uma ontologia OWL Full válida.

```
<ResearchLab rdf:ID="Mobilab_Unisinos">
  <NetworkType rdf:datatype="xsd-string">
  802.11b
  </NetworkType>
  <NetworkStatus rfs:datatype="xsd-string">
  Open
  </NetworkStatus>
</ResearchLab>
```

Figura 5. Representação em OWL DL de algumas informações sobre o Mobilab[2]

Um exemplo de representação de uma propriedade de uma instância de classe em OWL DL pode ser visto na **Figura 5**. Neste caso, são representadas informações sobre o tipo de rede disponível e o status da rede em um laboratório de pesquisa da Unisinos.

2.3.4 Ferramentas

Existem diversas ferramentas que permitem a modelagem ou o trabalho visual com as ontologias. O Protégé [32] é uma delas, permitindo a modelagem visual das ontologias. A **Figura 6** demonstra a utilização do Protégé com o OWLviz, que é um *plugin* que permite a visualizaçaõ gráfica das ontologias criadas no Protégé. Uma das grandes vantagens do Protégé é que ele permite exportar as ontologias criadas em um grande número de formatos, incluindo RDF. Outra ferramenta interessante é o SemWeb [58], que é um conjunto de bibliotecas de código-fonte aberto utilizadas para manipular e persistir ontologias RDF em formato XML. O SemWeb foi empregado durante a etapa de implementação, para facilitar a manipulação de ontologias no *framework* .NET.

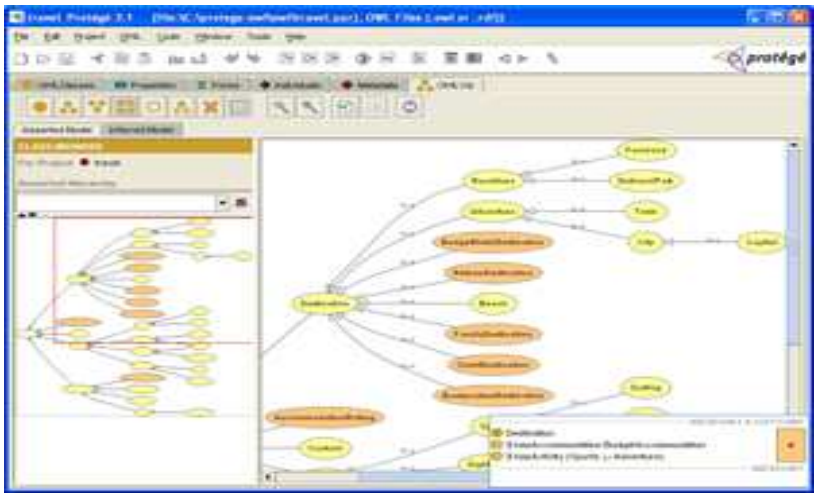

Figura 6. Captura de tela no Protégé OWL (usando o OWLviz)

[2] http://www.inf.unisinos.br/~mobilab

2.4 Similaridade

Similaridade pode ser definida como “o grau de relacionamento entre duas entidades” [18], ou seja, o quanto duas entidades quaisquer compartilham das mesmas propriedades ou recursos. A determinação da similaridade entre dois conteúdos quaisquer é um problema importante em diversas áreas de pesquisa, como a mineração de dados, a publicidade na mídia e o alinhamento de grupos especialistas. Esta determinação de similaridade pode ocorrer de duas formas distintas: primeiramente, através da comparação de termos em relação ao vocabulário, incluindo palavras e expressões (similaridade léxica [16]); também é possível buscar similaridades nos conceitos em si, ou seja, o significado das palavras e sentenças (similaridade semântica [17]). Pode-se perceber claramente que enquanto a similaridade léxica busca equivalência de símbolos, a similaridade semântica busca equivalência de idéias. Nas subseções a seguir serão apresentadas algumas técnicas para determinação de similaridade.

2.4.1 Similaridade Léxica

Matematicamente, a determinação de **similaridade léxica** entre conjuntos de dados é comumente tratada como um problema de métrica de *strings* [16]. O objetivo, neste caso, é obter uma pontuação que indique o quão parecidas são duas sequências de caracteres. O significado desta pontuação depende do método empregado no *matching* das sequências de caracteres, e é comumente representado por um valor entre 0 (nenhuma similaridade) e 1 (equivalencia total).

Existem diversos métodos para a determinação da similaridade léxica entre sequências de caracteres. A métrica chamada Distância de Levenshtein [37], também chamada de distância de edição (*edit distance*) mede a diferença entre duas sequências de caracteres, com base no número de operações necessárias para transformar a primeira na segunda. As operações permitidas são:

a) Inserção: adição de caracteres à sequência. Ex. **r**ato → **p**rato.
b) Remoção: exclusão de caracteres da sequência. Ex. **r**ato → ato.
c) Substituição: troca de um caractere por outro. Ex. **r**ato → **g**ato

Dessa forma, a Distância de Levenshtein entre as sequências “**pr**ato” e “ato” é de 2 (devido à realização de duas operações de remoção). A similaridade entre dois termos, neste caso, é diretamente proporcional ao número de substituições necessárias para transformar um termo em outro.

Na busca com o uso de lógicas difusas (*fuzzy search)* [39], é possível estabelecer relações de correspondências entre termos ou palavras com grafia similar. As técnicas baseadas em busca difusa normalmente operam através da redução de palavras ao seu termo raiz (*stemming*) [40], buscando a correspondência entre formas diferentes de uma mesma palavra ou termo. *Stemming* [41,42], também chamada de Radicalização, é uma técnica que procura reduzir variações de uma palavra aos seus radicais (base ou forma raiz de uma palavra). Por exemplo, as palavras “pesquisar”, “pesquisador” e “pesquisando” podem ser todas reduzidas ao termo radical “pesquisa”. Isso pode ser utilizado para calcular a similaridade entre termos ou conjuntos de termos. Deste modo, a busca *fuzzy*, quando aplicada a um texto, retorna palavras cuja sequência de caracteres seja similar à sequência procurada. Os algoritmos de radicalização, de uma forma geral, dependem do idioma utilizado [42], o que dificulta sua utilização na determinação de similaridades em grandes conjuntos de dados heterogêneos. Um dos algoritmos mais utilizados para na radicalização é o de Porter [42,43], que vem sendo desenvolvido há mais de 20 anos e possui versões adaptadas para diversos idiomas [42].

A busca *fuzzy* encontra similaridade entre sequencias de caracteres através da introdução de mutações em relação à sequência de caracteres de entrada, adicionando, removendo ou trocando caracteres, de modo a buscar correspondências aproximadas. Essas mutações obedecem ao mesmo grupo de operações definidas conforme a distância de Levenshtein, com a adição da operação de transposição, onde ocorre a troca de posição entre dois caracteres adjacentes (ex. **ra**t**o** → **ar**to). Após essa etapa, o número de operações realizadas na sequência de caracteres é computado, e serve como métrica para determinar o grau de similaridade entre os termos. Assumindo que o custo escolhido para todas as operações seja o mesmo, a similaridade é diretamente relacionada ao número de operações necessárias. Uma das aplicações mais comuns desse tipo de busca é a criação de verificadores ortográficos, com a sugestão de termos apropriados.

Outro tipo de medida de similaridade léxica é a medida de Similaridade do Cosseno (*cosine similarity*) [40], que retorna a similaridade entre dois vetores de n-dimensões. Isto é feito através da determinação do cosseno do ângulo entre estes (θ). Esta medida é descrita pela seguinte fórmula, onde A e B são os vetores:

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}.$$

Os valores dos vetores, no caso de sequências de caracteres, são as frequências dos termos dentro dos documentos analisados (tf-idf – *term frequency – inverse document*

frequency). O tf-idf é o produto entre a frequência de um termo no documento de entrada (o número de vezes que um termo aparece em um documento) e a frequência inversa no universo de documentos analisados (o logaritmo da divisão entre o número de documentos e quantos deles apresentam o termo pesquisado). Isso é feito de modo a dar a mesma "chance" para documentos de tamanhos diferentes. Por exemplo, em um documento de **3000** palavras, um determinado termo aparece **12** vezes. Sendo assim, a frequência deste termo no documento pode ser expressa como **(12 / 3000) = 0,004**. Assumindo um universo de **500** documentos, no qual esta mesma palavra apareça em **45** deles, a frequência inversa é calculada por **log (500 / 45) = 1.0457**. Desta forma, o tf-idf é dado como **0,004 x 1.0457 = 0,0041828**. Essa medida retorna um resultado entre **0** (extremamente oposto) e **1** (totalmente similar), já que não se pode ter tf-idfs negativos.

2.4.2 Similaridade Semântica

A determinação da **similaridade semântica** entre sequências de caracteres se dá através da determinação, de alguma forma, da proximidade do significado dos termos analisados. Isso pode ser feito através da análise de ontologias, que são representações formais do conhecimento através de uma rede de conceitos e de relacionamentos entre estes.

Os métodos usados na determinação da similaridade léxica podem ser empregados para facilitar a obtenção de uma métrica de similaridade semântica, ao encontrar equivalências entre os termos analisados e os conceitos presentes em uma ontologia. A busca por equivalências geralmente se vale de técnicas como o mapeamento taxonômico, que compara a similaridade entre duas taxonomias [44]. Isso é possível porque, conforme previamente mencionado, toda ontologia é também uma taxonomia. Mais uma vez, a pontuação varia comumente entre **0** (nenhuma similaridade) e **1** (similaridade total). Os trabalhos expostos na no **Capítulo 3** tratam em maiores detalhes de diversas abordagens para detecção de similaridades entre conjuntos de dados, com foco na similaridade semântica.

2.5 Considerações Sobre o Capítulo

Este capítulo discutiu os conceitos de computação ubíqua, trilhas e ontologias, bem como apresentou técnicas para determinação de similaridade entre conjuntos de dados. No **Capítulo 3** serão apresentados alguns trabalhos relevantes no âmbito da arquitetura proposta.

3 TRABALHOS RELACIONADOS

Nas seguintes subseções, serão analisadas três abordagens relacionadas à determinação de similaridades entre conjuntos de dados. Cada uma delas apresenta um enfoque diferente. A **Seção 3.1** descreve um método de obtenção de uma medida de similaridade semântica entre perfis de usuários baseando-se em informações auxiliares coletadas segundo um processo de "expansão" de termos através de ontologias, denominado *spreading*. A **Seção 3.2** descreve um conjunto de métricas para a formalização de medidas de similaridade entre instâncias de ontologias, tanto léxica quanto semântica, bem como apresenta um conjunto de experimentos realizados para validação dessas métricas. Por fim, a **Seção 3.3** descreve um trabalho que propõe uma arquitetura baseada em agentes para criar software orientado a trilhas.

3.1 Computing Semantic Similarity Using Ontologies

O trabalho de Thiagarajanm Manjunath & Stumptner (2008)[3] [18] trata da determinação da similaridade semântica entre conjuntos de palavras que descrevam duas entidades quaisquer. O trabalho propõe a utilização de um método baseado em *spreading activation networks* [45] com vários mecanismos de ativação (baseada em conjuntos e baseada em grafos) e *matching* de conceitos através de grafos bipartidos [46]. O trabalho é direcionado para a comparação de perfis de usuários de modo a determinar áreas de intersecção entre os interesses declarados nestes perfis. Os autores afirmam que o método desenvolvido pode ser empregado em qualquer cenário, com qualquer tipo de conteúdo.

O trabalho age sobre entidades representadas no formato BOW (*Bag of Words*) [47], em *frameworks* de recuperação de informações (*Information Retrieval Frameworks* – IRFs). BOW é um conjunto de termos com pesos associados que descrevem uma entidade. Isso é feito de forma que a similaridade entre duas entidades possa ser computada através dos pesos associados a cada termo.

As descrições das entidades (*Entity Descriptions* – ED) são criadas a partir do *spreading* de termos de uma ontologia no BOW, levando à criação do que os autores chamaram de BOW (*Bag of Concepts*).

---

[3] Desenvolvido como um projeto de pesquisa no centro de Pesquisa em Computação Avançada da *University of South Australia*, em parceria com a *Hewlett-Packard Development Company L.P.*

A medida de similaridade do cosseno [40] é o cálculo do ângulo do cosseno entre dois vetores, cada um deles modelado através da representação das entidades (BOW), e é calculado pela seguinte fórmula:

$$sim_{cos}(e_j, e_k) = \frac{\overrightarrow{V}(e_j) \cdot \overrightarrow{V}(e_k)}{|\overrightarrow{V}(e_j)||\overrightarrow{V}(e_k)|},$$

onde $\overrightarrow{V}$(ej) é a representação da entidade ej e a distância euclidiana $(|\overrightarrow{V}(e_j)|)$ da entidade ej é $\sqrt{\sum_{i=1}^{n} w_i^2}$.

Spreading é o nome do processo através do qual novos termos são adicionados à descrição de uma entidade, através da vinculação de uma ontologia. O trabalho dos autores consistiu em aperfeiçoar este método, propondo um sistema de métricas para a determinação da similaridade com redes de ativação de *spreading* baseadas em ontologias. No caso da pesquisa analisada, várias estratégias de *spreading* foram testadas, através de diversas ontologias. Duas destas foram a Wordnet [48] e a Wikipedia [49]. O processo de *spreading* é controlado por dois parâmetros básicos: tipos de relacionamento e função-peso. Dessa forma, um *spreading* baseado na ontologia da Wikipedia pode ser limitado apenas à categoria-pai imediatamente superior ao termo pesquisado. Além disso, um conjunto de funções-peso define quais os pesos para cada um dos tipos de relacionamento permitidos. Os tipos de relacionamentos expressam de que maneira o termo obtido na ontologia durante o *spreading* é associado ao termo original, e correspondem aos valores dos pesos. Por exemplo, se tomarmos o termo *músico*, o *spreading* através da Wikipedia pode conduzir a termos como *artista*, com um relacionamento do tipo *ParentOf*. Por sua vez, se tomarmos como ponto de partida a Wordnet, e considerarmos apenas relações de sinônimo/antônimo, outros termos serão associados (e no caso dos antônimos, o peso será negativo, para simbolizar a distância no relacionamento entre os conceitos).

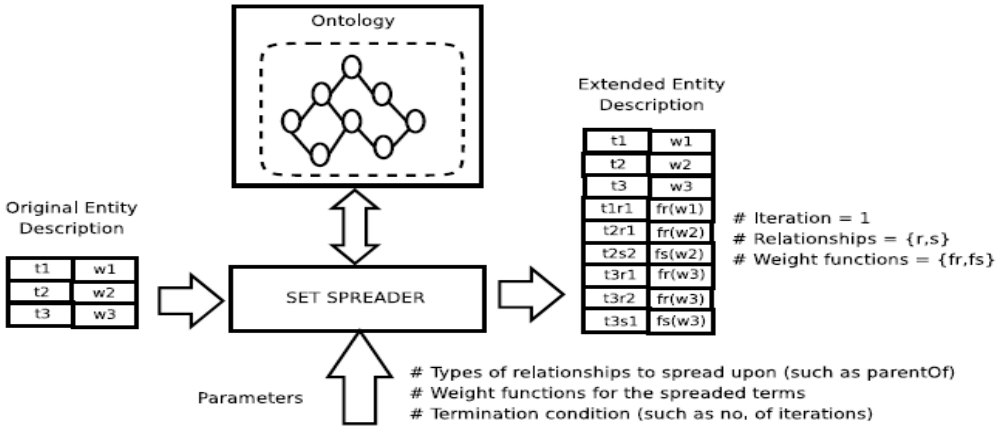

Figura 7. Processo de *spreading* baseado em conjuntos (conforme [18])

Os autores propuseram duas estratégias de *spreading*. Uma delas baseia-se em conjuntos (**Figura 7**), de modo que os termos alcançados sejam adicionados na ED original. Essa estratégia agrega termos à ED enquanto um número predeterminado de iterações não for alcançado ou então quando não existirem mais termos relacionados.

A segunda estratégia de *spreading* baseia-se em grafos (**Figura 8**), e tem por objetivo criar uma rede semântica [50]. Esta rede semântica é posteriormente utilizada para estabelecer relacionamentos entre as entidades que se deseja analisar. O processo começa com um grafo vazio. Em seguida, todos os termos presentes na ED são adicionados no grafo. Uma lista de nodos "abertos" é criada. Para cada termo, este é adicionado na lista de nodos abertos, e os sucessores são determinados segundo as relações presentes na ontologia empregada, sendo que para cada sucessor obtido, uma aresta é criada entre este e o termo pai. Cada novo termo adicionado entra na lista de abertos, e o processo se repete até que a condição de parada seja alcançada (aqui os autores também empregaram o número de iterações). A diferença entre as duas estratégias de *spreading* é que na estratégia em grafos as relações entre os termos são preservadas, tornando-se arestas no grafo obtido.

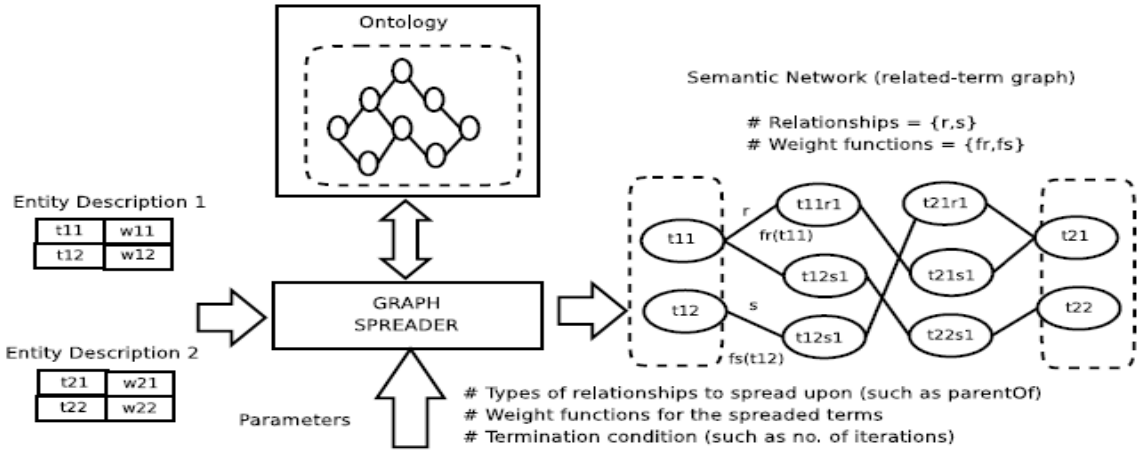

Figura 8. Processo de *spreading* baseado em grafos (conforme [18])

Para exemplificar a aplicação da técnica, os autores sugerem o seguinte exemplo, extremamente simplificado: dois usuários de um sistema de computador possuem perfis individuais correspondentes. Um dos perfis possui o termo *google* associado, enquanto que o outro possui o termo *yahoo*. Deseja-se determinar a similaridade entre estes dois perfis.

```
– e1 = {<google, 1.0>}, e
– e2 = {<yahoo, 2.0>}.
```

Utilizando uma comparação através da medida de similaridade do cosseno, obtém-se um conjunto vazio, o que indica ausência de similaridade entre os dois termos. No entanto, se um avaliador humano fosse atribuir uma medida de similaridade entre estes dois temos, esta certamente seria maior do que zero, porque este consideraria o relacionamento existente entre o significado dos termos (ambos são provedores de serviços de busca e oferecem email, por exemplo). Ao empregar a técnica proposta pelos autores, analisando o mesmo exemplo, pode-se perceber a utilidade/efetividade do processo de *spreading* na determinação de similaridade entre dois conceitos. Um exemplo é a análise da categoria-pai dos termos conforme estão dispostos na Wikipedia:

- e01 = {<google, 1.0>, <internet search engines, 0.5>}, e
- e02 = {<yahoo, 2.0>, <internet search engines, 1.0>}.

A intersecção dos conjuntos, nesse caso, é o conjunto dado por $(e'_1 \cap e'_2 \neq \emptyset)$, indicando uma certa similaridade (agora a similaridade do cosseno é 0,2). Isso demonstra que o *spreading* garante que seja detectada alguma similaridade entre as descrições das entidades. Neste caso, foi empregada a estratégia de *spreading* baseada em conjuntos de termos, com tipo de relacionamento igual a *ParentOf* (ambos google e yahoo são instâncias de *internet search engines*).

Os autores validaram a abordagem proposta através de um estudo onde foram envolvidos 10 usuários. Foi solicitado que estes relacionassem uma lista de 10 documentos que descrevessem sua área de pesquisa/atuação, bem como um conjunto de palavras chave descrevendo cada um destes documentos. Essas informações foram usadas para gerar **perfis estáticos** para os usuários, através de uma ferramenta apropriada. Em seguida, os perfis foram compartilhados, e foi pedido que os usuários definissem um nível de similaridade entre cada um dos 45 pares únicos de perfis, variando entre [0..100] (sendo que 100 representa similaridade total). Para cada um dos perfis obtidos, foram gerados perfis baseados na Wordnet e na Wikipedia. Para cada um dos 45 pares únicos de perfis gerados, a similaridade entre estes foi computada. De maneira geral, os resultados se apresentaram favoráveis, conforme demonstrado na **Figura 9**, onde a similaridade entre dois perfis foi calculada segundo as métricas *Similaridade do Cosseno* (em amarelo) e *Spreading baseado em Grafo* (em roxo).

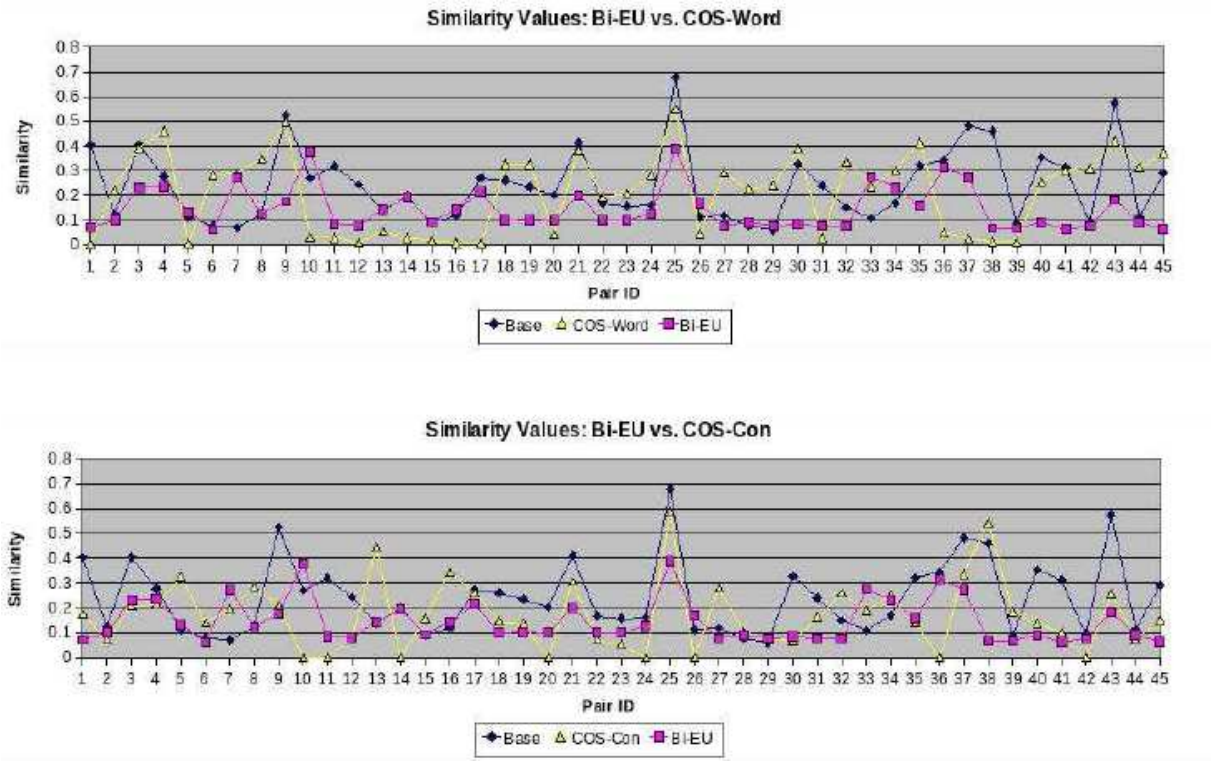

Figura 9. Análise comparativa entre dois perfis de usuário (conforme [18])

Outro teste conduzido pelos autores foi o de monotonicidade, buscando comprovar se a similaridade entre duas EDs que sofreram *spreading* aumentava ou diminuia conforme o número de iterações permitidas durante o processo de *spreading*. A **Figura 10** mostra que o *spreading* baseado na Wikipedia tende a aumentar o valor da similaridade conforme cresce o número de iterações. A desvantagem é o aumento no tamanho do perfil de acordo com o número de iterações, o que pode ser justificável dependendo do tipo de aplicação que se deseja implementar.

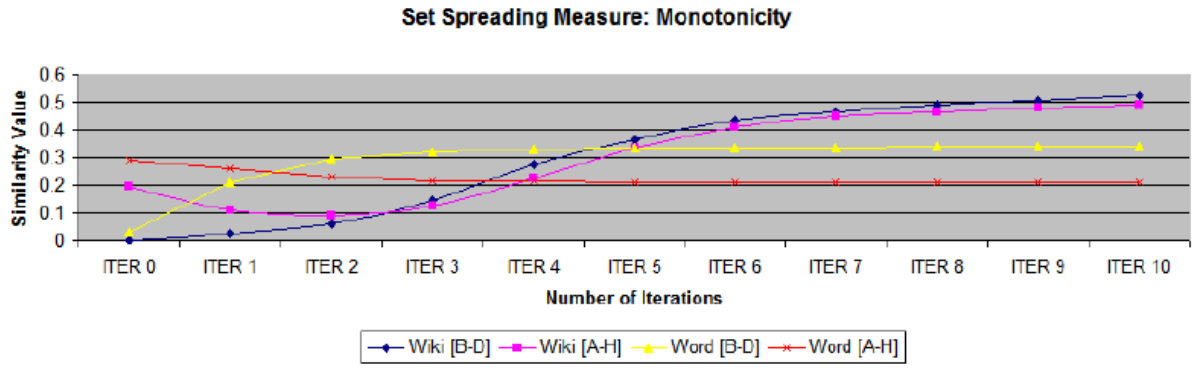

Figura 10. Análise de monotonicidade entre dois perfis de usuário (conforme [18])

Os autores afirmam que a mesma técnica pode ser utilizada em associação com diversas métricas baseadas em pesos atribuídos a termos (por exemplo, medida de similaridade do cosseno e coeficiente de Dice [51] em aplicações como mapeamento de

ontologias, *clustering* de documentos e matchmaking para jogos online, apenas para citar alguns exemplos. De acordo com os autores, "(…) [essa abordagem] demonstrou um aumento no grau de certeza em relação às outras analisadas [pelo grupo de pesquisa], inclusive quando comparada com similaridade determinada por meios humanos (…)".

Pode-se dizer que a contribuição fundamental do trabalho dos autores foi estender a noção de similaridade semântica entre entidades, passando a considerar relacionamentos entre os termos que constituíam o BOW. Isso foi realizado através do *spreading*. Além disso, essa metodologia possibilita que seja detectada uma similaridade entre dois usuários cujos perfis contenham termos que não possuam nenhuma relação léxica (como no exemplo das ferramentas de busca). Outro ponto forte é que ao empregar ontologias na classificação, problemas como a polissemia (uma palavra com mais de um significado, dependendo do contexto) são evitados. Um exemplo dado pelos autores são os termos *jaguar* (animal, modelo de carro, sistema operacional) e *apple* (maçã, gravadora, fabricante de *hardware* & *software*).

Como passos futuros os autores exploram técnicas que permitam decidir automaticamente a melhor estratégia de *spreading* baseando-se no conteúdo da ED, determinando inclusive a melhor ontologia para buscar os termos associados.

3.2 Measuring Similarity Between Ontologies

Maedche & Staab [19] apresentam um conjunto de métricas de similaridades para ontologias e um método de avaliação empírica dos resultados. A motivação para tal foi a falta de um método formal para a determinação da similaridade entre ontologias que descrevam entidades no mundo real. Em um primeiro momento, os autores desmembram o problema, produzindo uma série de métricas que podem ser utilizadas para a determinação da similaridade entre ontologias, tanto no nível léxico quanto no nível conceitual (semântico). De maneira geral, estas métricas descrevem a intersecção entre dois conjuntos de conceitos dispostos sob a forma de ontologias. Logo após, os autores apresentam um conjunto de experimentos realizados, tomando como base as métricas produzidas anteriormente.

Na primeira parte, é definida uma visão em dois níveis do conceito de ontologias, buscando separar o nível léxico do nível semântico. Para isso, os autores estabelecem expansivamente uma série de conceitos básicos e funções auxiliares. Um dos exemplos de métricas adotadas para a determinação de similaridade entre ontologias ao **nível léxico** é a Distância de Levenshtein (vista na **Seção 2.4**), que fornece o número mínimo de operações (inserção, remoção e substituição de *tokens*) que devem ser realizadas em uma determinada sequência para transformá-la em outra. Os autores fornecem um exemplo (aqui estendido):

Dados os termos “Top Hotel”, “Top_Hotel” e “Top hotels”:
ed(“Top Hotel”,”Top_Hotel”) = 1;
ed(“Top Hotel”,“Top hotels”) = 2;
ed(“Top_Hotel”,”Top hotels”) = 3;

Baseando-se na distância de Levenshtein, os autores propõem uma medida de similaridade léxica (*String Matching* – SM) que compara duas entradas léxicas Li e Lj:

$$\mathrm{SM}(L_i, L_j) := \max\left(0, \frac{\min(|L_i|, |L_j|) - \mathrm{ed}(L_i, L_j)}{\min(|L_i|, |L_j|)}\right) \in [0, 1].$$

Essa medida retorna um grau de similaridade entre [0..1] (onde 1 significa correspondência total e 0 significa uma correspondência ruim). A medida considera o número de operações que devem ser realizadas para transformar uma sequência de caracteres em outra, dividindo pelo tamanho da menor *string* comum. Conforme o exemplo anterior:

Dados os termos “Top Hotel”, “Top_Hotel” e “Top hotels”:
SM(“TopHotel”,”Top_Hotel”) =7/8;
ed(“Top Hotel”,“Top hotels”) = 6/9 = 2/3;
ed(“Top_Hotel”,”Top hotels”) = 6/9 = 2/3;

Como exemplo de determinação de similaridade entre ontologias no **nível conceitual**, tem-se o mapeamento taxonômico (*Taxonomic Overlap* – TO), que compara a similaridade entre duas taxonomias. A medida *semantic cotopy* (SC) define o conjunto de todos os conceitos pai e filho de um termo em uma taxonomia. Por exemplo:

$$SC(C_i, H) := \{C_j \in A | H(C_i, C_j) \vee H(C_j, C_i)\},$$

onde H(Ci, Cj) representa Ci como sendo um subconceito de Cj (A é um conjunto de conceitos da ontologia). Isso quer dizer que existe uma entrada léxica **L** que se refere a dois conceitos de duas taxonomias diferentes. Sendo assim, a medida SC compara termos de uma ontologia fonte contra uma ontologia alvo. A **Figura 11** exemplifica a comparação parcial da similaridade entre duas taxonomias. Nela, o *taxonomy overlap* TO’(“hotel”, H1, H2) é determinado pelas funções :

F_1^{-1}*(SC(F({“hotel”}), H1)) = {“hotel”, “accomodation”}*, e

F_2^{-1}*(SC(F({“hotel”}), H2)) = {“welness hotel”,”hotel”}*, que determinam a ligação entre conjuntos de entradas na série de termos da taxonomia.

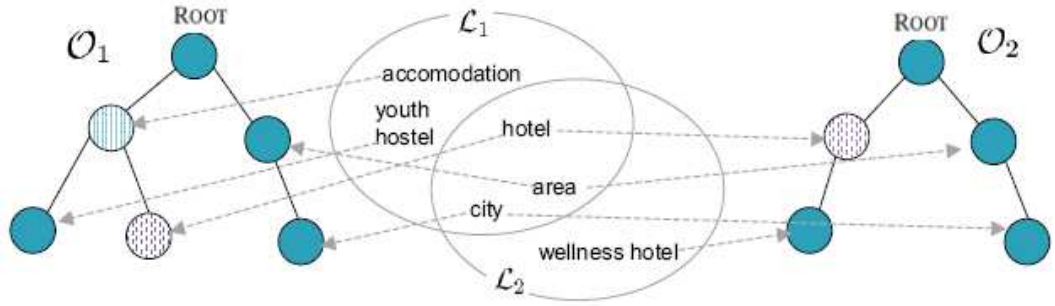

Figura 11. Duas ontologias de exemplo (conforme [19])

De modo a validar as métricas, foi conduzido um estudo durante um seminário realizado no instituto ao qual os autores são filiados. O tema do seminário foi a engenharia (criação) de ontologias. Os principais objetivos dos testes foram: i) avaliar a qualidade das métricas; ii) investigar o nível de similaridade entre ontologias tratando sobre o mesmo domínio, porém modeladas por pessoas diferentes. Durante os experimentos, foi solicitado a cinco pessoas (quatro "novatos" e um "*expert*" na área de criação de ontologias) que modelassem ontologias descrevendo um domínio real, amplamente conhecido. O ponto de partida foi um conjunto de textos-base que descreve o domínio, de modo a nivelar o conhecimento utilizado pelos participantes. As ontologias geradas durante esta etapa serviram como ponto de partida para um estudo empírico baseado nas métricas obtidas anteriormente.

De acordo com os autores, os avaliadores humanos concordaram entre si nas avaliações muito mais no que diz respeito aos conceitos em si do que nas relações entre estes. A **Tabela 1** apresenta os valores de SM para um dos diversos conjuntos de resultados obtidos, neste caso, uma ontologia que descreve conceitos relacionados à área de Turismo/Viagens.

Tabela 1. Valores típico para correspondência de strings

L_1	L_2	SM$(L1, L2)$
Sehenswuerdigkeit [seesight]	Sehenswürdigkeit [seesight]	0.875
Verkehrsmittel [vehicle]	Luftverkehrsmittel [air vehicle]	0.71
Zelt [tent]	Zeit [time]	0.75
Anzahl_Betten [number_beds]	hat_Anzahl_Betten [has_number_beds]	0.77

Uma observação interessante é que valores SM (*string matching*) acima de 0,75 representam, em geral, "boas" correspondências. Isso não significa que não houve problemas. Por exemplo, o valor para *Zelt* ("cobertura") e *Zeit* ("tempo") ficou no limiar de uma boa

correspondência, o que é de se esperar de uma fórmula que leve em conta apenas a equivalência léxica.

Os autores esperam aplicar as idéias desenvolvidas nesse trabalho criando uma ferramenta de busca de ontologias, capaz de informar um grupo de ontologias semelhantes a uma ontologia base, fornecida pelo usuário. Além disso, com base no que já foi desenvolvido, é possível descobrir mapeamentos entre ontologias, de modo a facilitar sua criação em atividades colaborativas.

3.3 *TrailTRECer*

TrailTRECer [20] é um projeto que reúne uma arquitetura baseada em agentes e um modelo aberto de dados orientado para o desenvolvimento de aplicações baseadas em trilhas. O *framework* formado pelo TrailTRECer define que o objetivo principal dessas aplicações é assistir os usuários na navegação entre conteúdos ou na busca de informações específicas. Dessa forma, a funcionalidade principal do TrailTRECer é possibilitar que um usuário "siga" a trilha de navegação de outro usuário, facilitando a busca de informações. O framework é adaptável de modo a possibilitar a integração de diferentes aplicações.

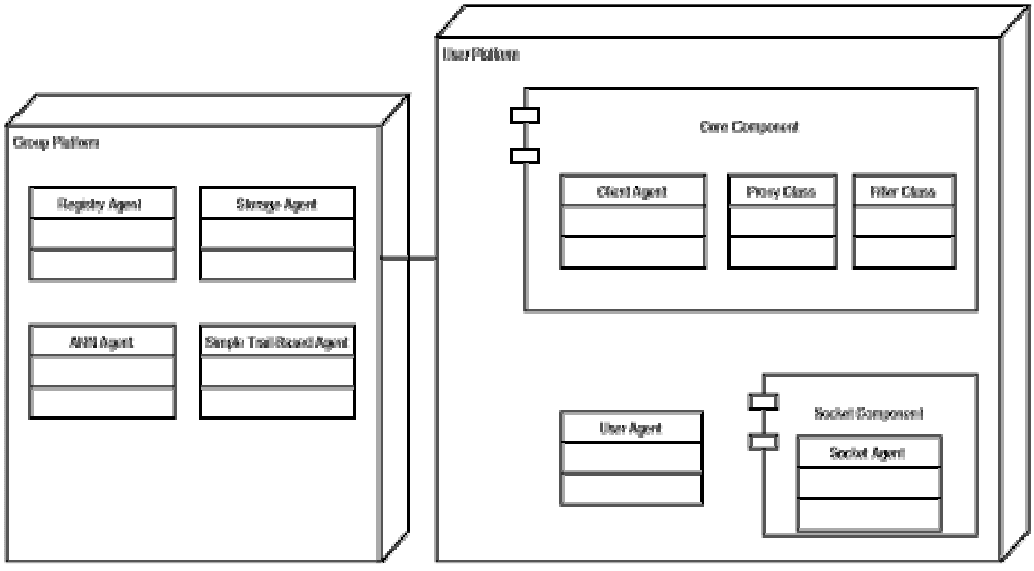

Figura 12. Arquitetura do TrailTRECer

A arquitetura do TrailTRECer segue o modelo FOHM [52] (*fundamental open hypermedia model*), que é um modelo comum de dados capaz de expressar dados de trilhas. A **Figura 12** exibe o *framework* TrailTRECer, mostrando quais partes podem ser adaptadas às necessidades do usuário/aplicação em questão. A arquitetura é dividida em duas partes: 1) *User Platform* (realiza a aquisição de trilhas, através dos agentes *Socket* e *Client* e possibilita a visualização das informações através do agente *User*); 2) *Group Platform* (realiza

o processamento e armazenamento de trilhas). A comunicação entre os agentes é possível através de um outro agente, chamado *Registry*.

O modelo foi desenvolvido em Java, e a implementação-base da arquitetura dos agentes foi realizada através do framework SoFAR [53] (*the Southampton Framework for Agent Research*). A validação ocorreu através da criação de três modelos de integração envolvendo três aplicações de exemplo diversas (um cliente de impressão, um sistema de trilhas para documentos do Office e um cliente de navegação assistida para a web). Através destas aplicações, o modelo é capaz de adquirir dados sobre atividades específicas sendo realizadas e visualizar documentos relacionados a cada atividade.

O trabalho desenvolvido no TrailTRECer é interessante, no sentido de demonstrar a aplicabilidade do modelo FOHM e da abertura da informação para aplicações baseadas em trilhas. Como trabalhos futuros, os autores procurarão focar na captura de trilhas em cenários de rede interna (*intranets*), como diretórios compartilhados, e desenvolver outros agentes de modo a sofisticar o processamento das recomendações. Além disso, busca-se desenvolver a parte de algoritmos de similaridade e extração de tópicos dos materiais pesquisados.

3.4 Avaliação dos Trabalhos Pesquisados

A

Tabela 2 exibe um comparativo entre os trabalhos apresentados. Todos os trabalhos consistiram em abordagens para a determinação da similaridade semântica entre conjuntos de dados (através do uso de ontologias) ou então das próprias ontologias. Os critérios empregados nesse comparativo foram definidos em função da utilidade perceptível de cada alternativa analisada em relação à determinação de similaridade. Os critérios analisados foram os seguintes:

a) **Tipos de comparação possíveis:** quais tipos de dados a abordagem específica contempla para comparação;
b) **Mecanismo de comparação:** quais técnicas, algoritmos, recursos e/ou idéias centrais foram empregados que diferenciam este trabalho dos demais;
c) **Aplicabilidade real:** possibilidade de generalização dos métodos empregados; além disso, define o escopo em que a técnica pode ser aplicada;
d) Método de avaliação (aplicação): aplicações empregada na avaliação, ou seja, a maneira encontrada para testar e/ou validar a abordagem proposta;
e) **Ontologias empregadas:** enumera as ontologias nas quais cada trabalho baseou-se para a determinação de similaridade, sejam ontologias explícitas (segundo a

definição clássica) ou derivadas através de outros conjuntos de dados (Wikipédia, diretórios *web*, etc...).

Tabela 2. Comparativo entre abordagens de determinação de similaridade

	Thiagarajan, Manjunath & Stumptner 2008	**Maedche & Staab, 2002**	**TrailTRECer**
Tipos de comparação possíveis	Somente semântica	Léxica e semântica	Somente léxica
Mecanismo de comparação	Enriquecimento das inferências semânticas através do mecanismo de *spreading*	Visão em dois níveis, métodos distintos; Métrica própria baseada na distância de Levenshtein	Extração de informações e recomendação.
Aplicabilidade real	Específica, porém de fácil generalização (segundo os autores)	Quaisquer ontologias que descrevam entidades do mundo "real"	Prático, validado através de três aplicações de mundo "real"
Método de avaliação (aplicação)	Determinação do grau de similaridade entre perfis enriquecidos com *spreading*	Determinação do grau de similaridade entre um grupo de ontologias relacionadas	Recomendação de documentos, sistema de navegação web assistida, cliente de impressão.
Ontologias empregadas	WordNet, Wikipedia	Criadas pelos próprios usuários durante a etapa de validação	FOHM extendido para trilhas, comunicação entre os agentes

Como as pesquisas na área de trilhas são relativamente novas, foi difícil encontrar trabalhos relacionados ao processo de determinação de similaridade de dados componentes das trilhas. No entanto, todos estes trabalhos contribuíram, em maior ou menor grau, para a realização desta proposta. A técnica de *spreading* [18] pode ser empregada para determinar a similaridade tanto entre conceitos associados a eventos ocorridos em um determinado momento quanto à localização geográfica onde estes eventos ocorreram. No primeiro caso, a estratégia de *spreading* baseada em conjuntos seria ideal, pois não se pode garantir que todas as descrições de eventos em um contexto possuam correspondências em uma ontologia. Além disso, segundo os autores, ela é surpreendentemente robusta em casos onde não exista propriamente uma ontologia (no caso de esta ser gerada dinamicamente, através de busca na *web*, por exemplo). Mesmo assim, deve-se ter cuidado de definir um número máximo de iterações para a obtenção dos dados, bem como um timeout para o processo de *spreading*. No segundo caso (localização geográfica), a estratégia de *spreading* baseada em grafos parece ser a mais indicada, visto que os relacionamentos entre localidades podem facilmente ser extraídos de uma ontologia que descreva, por exemplo, bairros, cidades e regiões. Novamente, deve-se definir *timeout* e número máximo de iterações.

As contribuições do segundo trabalho relacionado [19] vieram em dois pontos. Por um lado, a separação da determinação de similaridades entre nível léxico e nível semântico, bem como os algoritmos escolhidos para cada uma, foi bastante esclarecedora. Além disso, em um segundo momento, os autores empregaram as técnicas apresentadas na determinação da similaridade entre ontologias. Esta proposta de dissertação assume que as trilhas estão representadas na forma de ontologias. Desta forma, o trabalho é interessante no sentido da descoberta de mapeamentos entre ontologias e grau de semelhança.

O terceiro trabalho [20] contribuiu no sentido de demonstrar uma instância de arquitetura baseada em agentes que integra o trabalho com trilhas. A arquitetura possibilita construir aplicações de propósitos gerais que utilizem trilhas para atividades como direcionar a navegação e facilitar o trabalho dos usuários ao buscar conteúdo, sendo que a determinação do relacionamento entre as trilhas (com base em suas informações componentes) fica a cargo das aplicações.

O foco deste trabalho é aliar a estruturação de serviços de cálculo de similaridade de trilhas a uma arquitetura de modularizada, possibilitando a criação de aplicações de propósitos genéricos. Assim sendo, espera-se amalgamar os pontos fortes dos trabalhos analisados, bem como contornar suas limitações, de modo a agregar funcionalidades para a criação de novas aplicações que utilizem trilhas.

3.5 Considerações sobre o Capítulo

Esse capítulo apresentou três abordagens para a determinação da similaridade entre conjuntos de dados, através do emprego de ontologias. Um ponto interessante mencionado em todos os trabalhos analisados é que essa determinação, quando se dá no nível semântico, é invariavelmente mais pesada em termos de poder computacional do que a determinação da similaridade no nível léxico. Ainda assim, a similaridade semântica é mais “útil”, no sentido que descreve melhor as relações entre os elementos, sem se perder em situações inconvenientes, como a polissemia (um termo, vários significados). O próximo capítulo apresenta uma arquitetura que reúne as idéias mais interessantes entre os trabalhos pesquisados. Esta arquitetura é voltada para a determinação do grau de similaridade entre trilhas genéricas de entidades, possibilitando uma ampla gama de aplicações.

4 SIM

Nesse capítulo é apresentada uma arquitetura que fornece os meios básicos para a determinação de similaridades em dados que descrevem trilhas de entidades genéricas. Será apresentada uma visão geral da arquitetura, descrevendo seus componentes. O SIM contempla o armazenamento e a detecção de similaridades sobre uma base contendo dados que descrevem trilhas de entidades. O SIM foi projetado para ser independente do sistema que o acessa, no entanto, trabalha com trilhas em um formato específico. Estas informações de trilhas devem estar estruturadas conforme uma ontologia, que varia de acordo com cada aplicação. Isso deve-se ao fato de que duas aplicações não possuem necessariamente os mesmos requisitos no que diz respeito ao armazenamento de informações e ao interesse em determinar similaridade sobre conjuntos específicos de dados. Dessa forma, os dados das trilhas devem obedecer a uma ontologia definida de acordo com as necessidades específicas de cada aplicação. A definição e/ou especificação dessa ontologia é de responsabilidade do analista ou projetista da aplicação, que determina quais informações são relevantes para cada aplicação em particular. A ontologia descreve tanto os eventos que compõem a trilha (data, duração, etc), bem como os locais onde estes mesmos eventos podem ser realizados. Esta ontologia possibilita representar regiões em diferentes níveis de composição hierárquica, isto é, um determinado local/região pode abrigar outros locais/regiões (geográficas ou não).

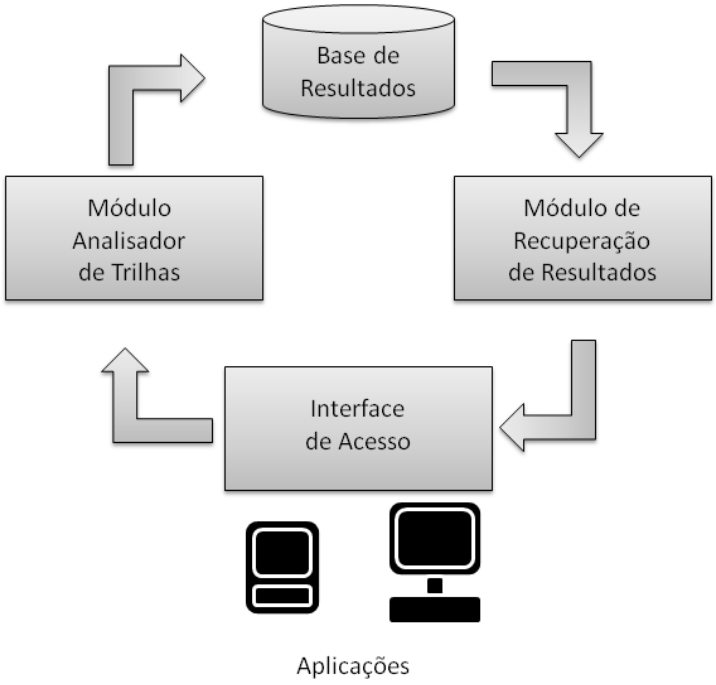

Figura 13. Visão geral da arquitetura

A arquitetura SIM é composta pelos seguintes módulos (**Figura 13**):

1) uma **Interface de Acesso**, que oferece o meio pelo qual as aplicações-cliente acessam e configuram as funcionalidades oferecida pelo SIM;

2) um **Módulo Analisador de Trilhas**, que realiza análise de similaridade nos conteúdos das trilhas;

3) um **Módulo de Recuperação de Resultados**, que possibilita a obtenção rápida de resultados já obtidos anteriormente, economizando tempo em análises repetidas;

As **Aplicações** conectam-se à **Interface de Acesso**. Os resultados chegam até as aplicações através do Módulo de **Recuperação de Resultados**. A arquitetura assume que, ao menos, os seguintes dados sejam disponibilizados para caracterizar uma trilha (**Tabela 3**):

Tabela 3. Formato básico assumido para as trilhas

Campo	**Subcampo**
ID da trilha	
*Lista de **eventos** (segundo definição na ontologia de eventos)*	*Data inicial*
	Data final
	Local (segundo ontologia de locais)
	Natureza do evento

As trilhas são informadas pelas aplicações-cliente no momento em que é solicitada a comparação de similaridade. Com base nas informações das trilhas, e de modo a estabelecer o coeficiente de similaridade, três tipos de dados são empregados:

a) Identificação dos locais visitados durante os deslocamentos efetuados pelas entidades;
b) Identificação dos eventos gerados, que são consequência das interações efetuadas pelas entidades dentro dos locais definidos pela aplicação;
c) Tempo de duração dos eventos e/ou permanência nos locais onde os eventos são gerados.

Todos estes dados são descritos conforme a ontologia citada anteriormente. Os módulos que compõem a arquitetura serão discutidos em detalhes nas próximas subseções.

4.1 Interface de Acesso

A **Interface de Acesso** serve como ponte entre as aplicações-cliente e a arquitetura propriamente dita. Esta é definida como um provedor de serviços de consulta, que opera

realizando consultas específicas, recebidas das aplicações-cliente. As consultas realizadas através da interface de acesso agem sobre dados informados pelas aplicações, como parâmetros. Estes parâmetros definem as trilhas que serão comparadas, os tipos de comparações que serão realizadas e os pesos que serão associados a cada tipo de comparação. A **Tabela 4** resume os serviços oferecidos pela **Interface de Consulta**. Com as chamadas enumeradas nesta tabela espera-se que seja possível suportar uma ampla gama de aplicações. As chamadas feitas através da **Interface de Acesso** são enviadas ao **Módulo Analisador de Trilhas**.

Tabela 4. Serviços oferecidos pela interface de consulta

Serviço	**Parâmetros**	**Retorno esperado**	**Descrição**
Sim (A,B)	- List<Trail> trails (A, B) - List<SIM_TYPE> simTypes - List<Weight> weights	Valor entre 0 e 1	Valor representativo do grau de similaridade entre as trilhas comparadas. O valor 0 indica nenhuma similaridade, 1 indica equivalência total.
NTrailsAlike	- Trail trail - Int numTrails - Float sim	List<Trail>	Lista de numTrails trilhas cujo grau de similaridade seja maior ou igual a sim
SetCache	- CacheStrategy cacheStrategy - Int numRegs	(Nenhum)	Configura a estratégia de *cache* selecionada, mantendo os últimos registros segundo a estratégia selecionada
SetOntology	- Ontologia em formato RDF	(Nenhum)	Configura a ontologia a ser usada para as próximas invocações da interface.

O SIM prevê que aplicações diferentes possuem necessidades diferentes. Assim sendo, deve-se decidir em quais níveis se dará a determinação de similaridade. Isso é necessário devido à natureza das aplicações que podem surgir. Por exemplo, no caso de uma aplicação de monitoração de frota desenvolvida por uma empresa de taxi (envolvendo trilhas de veículos), o tempo entre cada novo "ponto" de trilha e a duração dos mesmos são relevantes, já que trazem informações que podem ser utilizadas para determinar otimizações nas rotas, ou detectar engarrafamentos, por exemplo. Por outro lado, uma aplicação que analisasse trilhas de indivíduos em busca de informações sobre mobilidade social não tem utilidade para a duração dos pontos de trilha. Além disso, a primeira aplicação traria informações como a duração de cada corrida, autonomia do veículo, entre outros, enquanto que a segunda

aplicação levaria em conta primeiramente dados demográficos, como o tempo de serviço, idade, taxa de empregabilidade, região do país, dados da família, entre outros. Os tipos de comparações a serem realizadas são especificados quando da realização da chamada na **Interface de Acesso**. As aplicações-cliente devem informar em quais níveis da ontologia se dará a comparação. A **Tabela 5** traz os tipos de comparação suportados pela arquitetura.

Tabela 5. Tipos de comparação suportados

Tipo	**Descrição**
CMP_EVENT	Executa comparação nos campos que descrevem a natureza de eventos
CMP_LOCAL	Executa comparação nos campos que descrevem locais onde ocorrem os eventos das trilhas
CMP_INTERVAL	Executa comparações nos horários de início, fim e duração dos eventos
CMP_ALL	Efetua todas as comparações anteriores.

A associação de pesos para as operações de similaridade também é realizada na ocasião da chamada aos serviços disponibilizados pela **Interface de Acesso**. A soma de todos os pesos associados às operações deve ser igual a 1. No caso de serem informados valores cuja soma ultrapasse 1, os pesos individuais serão distribuídos uniformemente, de modo que a soma destes seja exatamente 1. O mesmo ocorre caso não sejam informados pesos, ou o valor informado para qualquer um destes seja negativo.

Todas as chamadas realizadas na **Interface de Acesso** são de natureza assíncrona; isto se deve ao fato de que a recuperação das trilhas através do **Módulo de Recuperação de Resultados** e subsequente comparação através do **Módulo Analisador de Trilhas** são processos potencialmente lentos. Desta forma, a **Interface de Acesso** não pode permanecer bloqueada enquanto aguarda pela conclusão de uma operação de similaridade. No melhor dos casos, a recuperação dos dados se dá através de um *cache* em memória, mantido pelo **Módulo de Recuperação de Resultados**. O cache é empregado para aproveitar o aspecto de localidade: se uma trilha é acessada em um determinado momento, provavelmente será acessada novamente em seguida. O cache faz com que as trilhas fiquem disponíveis por um pouco mais de tempo para recuperação rápida, acelerando o retorno de resultados para as aplicações.

Para cada comparação específica, existe apenas um resultado possível, uma vez que os algoritmos de comparação utilizados serão determinísticos. Dessa forma, é possível criar um identificador único (*hash*) a partir dos dados que compõem uma trilha (IDS, sequencias de

locais e eventos e outros parâmetros). De posse desse *hash* é possível saber se uma determinada comparação já foi realizada anteriormente pelo **Módulo Analisador de Trilhas**, e está pronta para ser recuperada através do **Módulo de Recuperação de Resultados**.

A **Figura 14** ilustra a composição de trilhas em dois níveis de locais. As localidades representadas nas trilhas, à direita, são simbolizadas por maiúsculas (A,B,C,D,X,Y,Z), enquanto que as localidades representadas dentro de A, à esquerda, são representadas por minúsculas (a, b, c, d). Event1, Event2 e Event3 representam os eventos que ocorreram quando a entidade "dona" da primeira trilha passou pela segunda vez pelo local C.

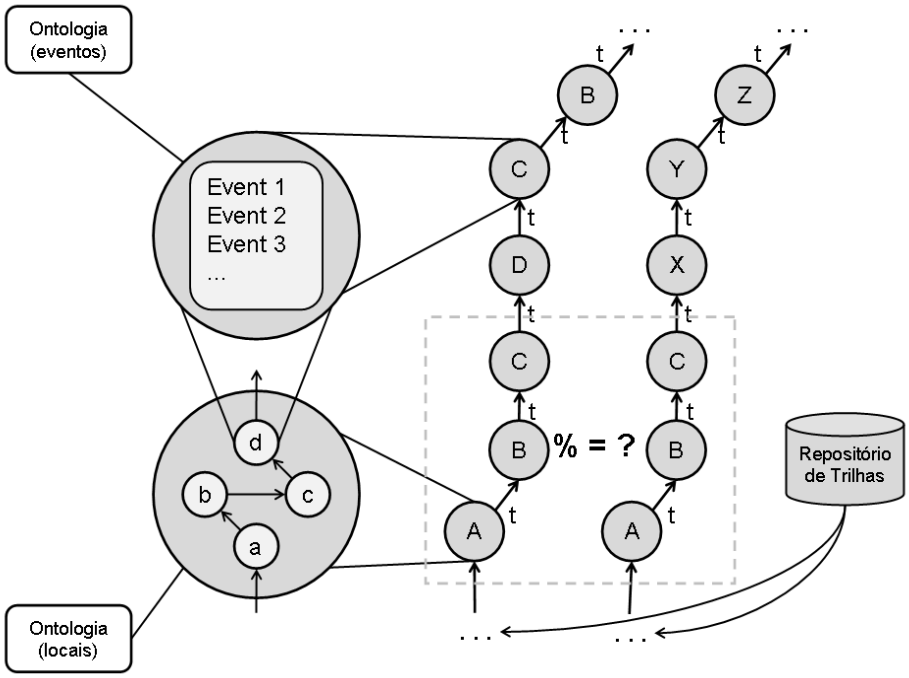

Figura 14. Exemplo de composição de trilhas

4.2 Módulo Analisador de Trilhas

O **Módulo Analisador de Trilhas** é o módulo da arquitetura responsável por fornecer os resultados das comparações de similaridade, conforme os parâmetros informados em uma requisição. Além disso, ele repassa os resultados das comparações para o **Módulo de Recuperação de Resultados**, que armazena as trilhas comparadas e os resultados em uma base de dados interna, bem como mantém um cache dos últimos resultados obtidos em memória, para recuperação rápida.

O **Módulo Analisador de Trilhas** processa as requisições sequencialmente, em ordem de chegada. Além da possibilidade de ser invocado explicitamente, o **Módulo Analisador de Trilhas** é proativo em disponibilizar as operações de similaridade das trilhas

mais recentemente comparadas (com base nos dados acessíveis na base de dados interna). Essas operações são realizadas para cada trilha que apareça no *cache* do **Módulo de Recuperação de Resultados** (isto significa que são trilhas que foram acessadas recentemente). Isto é feito de modo automático, como uma tarefa de baixa prioridade, não bloqueando o fluxo de execução. A **Figura 15** demonstra essa funcionalidade. 1) O **Módulo Analisador de Trilhas** busca a efetuação de todas as comparações possíveis de similaridade entre as trilhas que estão no *cache* do **Módulo de Recuperação de Resultados**. 2,3) Cada resultado é verificado quanto à sua existência na Base de Resultados; dessa forma, o **Módulo Analisador de Trilhas** não repete operações desnecessariamente. Assume-se que as operações de similaridade realizadas variam dependendo de quais partes das trilhas estiverem sendo comparadas, e do método empregado, fazendo com que surjam resultados diferentes. O **Módulo Recuperador de Resultados** contempla esta possibilidade automaticamente, uma vez que tais operações geram *hashes* diferentes quando aplicado um algoritmo de *hashing*.

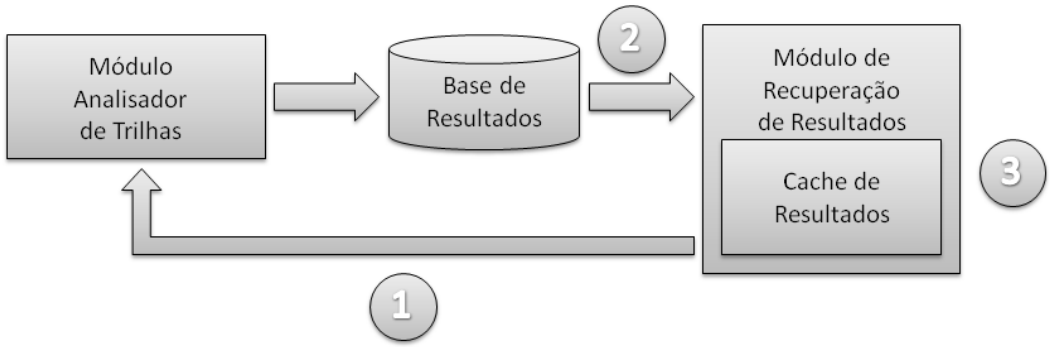

Figura 15. Módulo Analisador de Trilhas operando em modo automático

4.3 Algoritmos de similaridade

Nesta seção são apresentados os algoritmos empregados para a detecção de similaridade nos diferentes tipos de dados que compõem as trilhas. Estes algoritmos foram escolhidos de modo a facilitar a obtenção de um *score* de similaridade com base em métodos consagrados, condizentes com pesquisa efetuada na literatura da área.

4.3.1 Comparação de Eventos

Para a comparação entre os eventos gerados na trilha é usado o método de *spreading* [18], que consiste no uso de redes de ativação baseadas em grafos para a expansão da árvore de cada instância da ontologia escolhida para representar o universo de eventos possíveis. Isso torna possível representar equivalências entre termos das ontologias, mesmo que a grafia seja diferente, preservando a equivalência semântica. A **Figura 16** exemplifica o

método (sendo que o processo de *spreading* empregou a ontologia Wikipédia [49]). Em um primeiro momento, a comparação realizada parece ter resultado em um índice de similaridade de 0%, no entanto, através do processo de *spreading* controlado, nota-se que ambas pertencem à categoria “lista de universidades do sul do Brasil”.

```
– e1 = {<PUCRS, 1.0>},
– e2 = {<UNISINOS, 1.0>}.
```

```
– e01 = {<PUCRS, 1.0>, <universidades do sul do Brasil, 1.0>},
– e02 = {<UNISINOS, 1.0>, <universidades do sul do Brasil, 1.0>}.
```

Figura 16. Algoritmo *Graph spreading* aplicado a termos presentes nas trilhas

4.3.2 Comparação de faixas de tempo

A abordagem a ser utilizada para a determinação da similaridade entre o tempo transcorrido entre duas entradas de trilha é baseada no método de *cross-multiplication* [54]. Nesse método, o valor da similaridade é isolado através da multiplicação do numerador representativo da duração do deslocamento entre os pontos 1 e 2 de uma trilha A pelo mesmo deslocamento detectado em uma trilha B. No exemplo da **Figura 17**, onde as grandezas são dadas em minutos, o coeficiente de similaridade entre a duração dos deslocamentos de **A** até **B** é dado por:

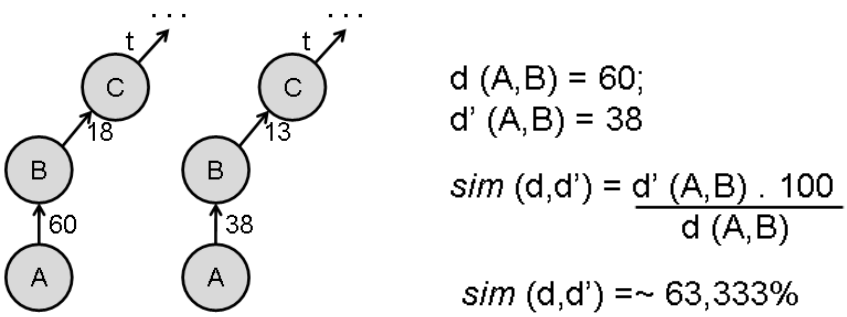

Figura 17. Determinação do coeficiente de similaridade de tempo

4.3.3 Sequências de Visitação (locais)

Para a comparação entre as sequências de visitação será usado o algoritmo chamado *tree edit distance* (distância de edição de árvore). Este algoritmo é baseado na distância de *Levenshtein* [37], retornando um coeficiente que indica o menor número de operações necessárias para transformar uma determinada árvore A em uma árvore B. Da mesma forma que o algoritmo de *Levenshtein* para *strings*, o *tree edit distance* lida com as operações de

inserção, remoção e substituição. Ao invés de caracteres, as operações se refletem sobre os nodos de uma árvore.

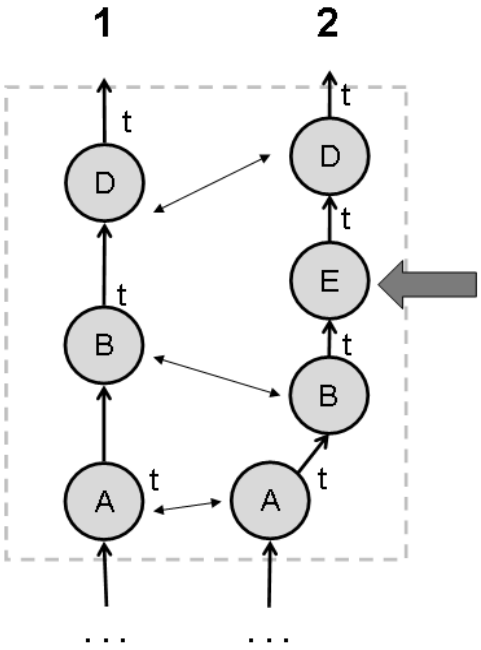

Figura 18. Algoritmo Tree Edit Distance aplicada às trilhas

No exemplo apresentado na **Figura 18**, apenas uma operação de inserção de nodo é necessária para transformar a árvore representada por A na árvore B. Analisando o exemplo de maneira superficial, isso leva a um índice de similaridade entre trilhas (geograficamente falando) de 0,75. No entanto, se levarmos em conta a composição de regiões mencionada anteriormente, as duas localidades visitadas representadas pela letra B podem ter abrigado percursos internos diferentes, conforme a **Figura 19**. Isso significa que visitas duplicadas aos locais devem ser mantidas, porque não se pode antecipar o nível de composição desejado ao consultar similaridades na base.

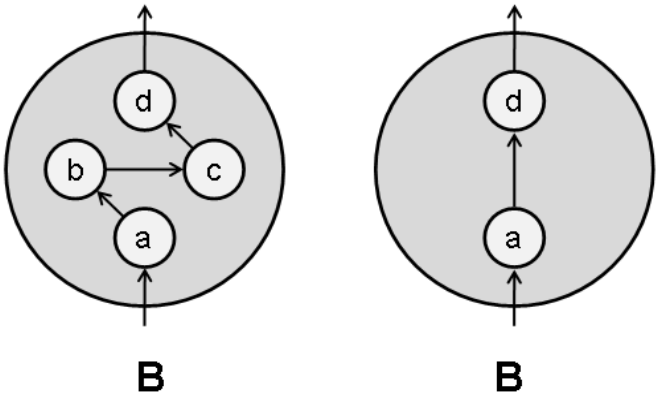

Figura 19. Diferenças no segundo nível de composição da trilha da Figura 18

Nesse caso, observa-se que a comparação baseada em *tree edit distance* deve ser propagada aos níveis inferiores da representação de locais. Uma maneira de representar isso é continuar a comparação de maneira recursiva, através do mesmo algoritmo, ajustando os pesos dos nodos em função do uso. Convenciona-se que a semelhança entre os dois percursos realizados dentro do contexto local de um mesmo nodo raiz valem 50% do valor daquele nodo (já que a composição garante que existam pelo menos dois nodos em um nível inferior dentro de um nodo raiz em um determinado nível). O peso é propagado para o nível superior, ajustando o valor do nodo raiz na determinação de similaridade.

4.4 Módulo de Recuperação de Resultados

O **Módulo de Recuperação de Resultados** é responsável por recuperar resultados de comparações diretamente da base de resultados. Além disso, este módulo também é responsável por guardar em uma base de dados interna os resultados das comparações realizadas pelo **Módulo Analisador de Trilhas**. Este módulo também implementa um *cache* de resultados de comparações já realizadas. Toda vez que uma comparação é solicitada sem que seus parâmetros mudem, o **Módulo de Recuperação de Resultados** pode retornar imediatamente uma cópia dos resultados residentes em memória. Isso é muito mais rápido do que recuperar os resultados da base, e é ideal no caso da aplicação exigir agilidade no retorno dos resultados. Um exemplo disso é uma aplicação de missão crítica em ambientes operados por máquinas, onde qualquer atraso no retorno dos resultados pode ter um impacto significativo.

A estratégia de *cache* de respostas do **Módulo de Recuperação de Resultados** é configurável pelas aplicações, através de uma chamada específica na **Interface de Acesso**. Dependendo do tipo de aplicação, pode ser mais interessante manter no *cache* os últimos resultados obtidos, ou então os resultados mais frequentemente acessados. A **Tabela 6** mostra as estratégias de *cache* suportadas.

Tabela 6. Estratégias de *cache* suportadas

Estratégia	**Repopulado?**	**Descrição**
CS_NENHUM	Nunca	Nenhuma estratégia de *cache*.
CS_ULTIMOS	A cada nova inserção	Últimos X resultados
CS_MAIS	Intervalo de tempo configurável	Os resultados com maior freqüência de solicitação por parte do Analisador
CS_HIBRIDO	Ambos os anteriores	

Na etapa de validação da arquitetura serão realizados testes de comparação de trilhas tanto com o *cache* ligado quanto desligado. Dessa forma, espera-se obter resultados relacionados à capacidade da arquitetura proposta de suportar uma grande quantidade de aplicações simultaneamente.

5 ASPECTOS DE IMPLEMENTAÇÃO E AVALIAÇÃO

Com base nas idéias descritas no **Capítulo 4**, um protótipo foi desenvolvido. Os componentes que formam a arquitetura foram implementados com tecnologias Microsoft. A linguagem escolhida para a implementação foi o C#, devido à familiaridade com a criação de aplicações no *framework* .NET. As ferramentas e aplicações produzidas foram executadas em ambiente MS Windows. Os componentes desenvolvidos serão detalhados nas seções subsequentes.

Com os serviços oferecidos pelo SIM é possível desenvolver diversos tipos de aplicações. Para validar a proposta, foi desenvolvida uma aplicação voltada à área de educação, possibilitando validar a arquitetura através de uma simulação. Esta aplicação consiste na integração do SIM com outros dois trabalhos. Em função de uma série de trabalhos já realizados no contexto da educação ubíqua, foi resolvido que este seria um dos assuntos focados nos testes.

A validação do SIM se deu através da integração deste com dois ambientes de suporte: o UbiTrail e o LOCAL. O UbiTrail [21] é um modelo voltado ao gerenciamento de trilhas de entidades em ambientes de computação ubíqua. Ele provê a representação de dados de entidades sob a forma de trilhas. O LOCAL (*Location and Context Aware Learning*) [23] é uma arquitetura de suporte à educação móvel e ubíqua voltada para ambientes em pequena escala. Esta arquitetura representa as informações sobre os usuários na forma de perfis individuais, sem a mesma sofisticação do UbiTrail. Além disso, o LOCAL possui um módulo chamado **Tutor**, que analisa continuamente os perfis dos usuários que estão em um mesmo contexto em busca de oportunidades pedagógicas, na forma de estímulo à interação e/ou recomendação de objetos de aprendizagem. Com base nestes dois ambientes, será montada uma solução capaz de oferecer um serviço de comparação de similaridade entre trilhas de entidades. Neste caso específico, as entidades do modelo são os alunos. Com os resultados da determinação de similaridade é possível formar grupos e recomendar material pedagógico observando os dados que compõem as trilhas dos usuários.

Além disso, será verificado o grau de impacto na implementação de *cache* em aplicações desse tipo. Será gerado um grande número de acessos aos serviços oferecidos pelo modelo, e isto será avaliado segundo diferentes estratégias de *cache*. Com o desenvolvimento do modelo e das aplicações de exemplo, espera-se obter resultados relacionados à utilidade das comparações, às estratégias utilizadas para minimizar o trabalho dos módulos, bem como a comparação com os trabalhos existentes na área.

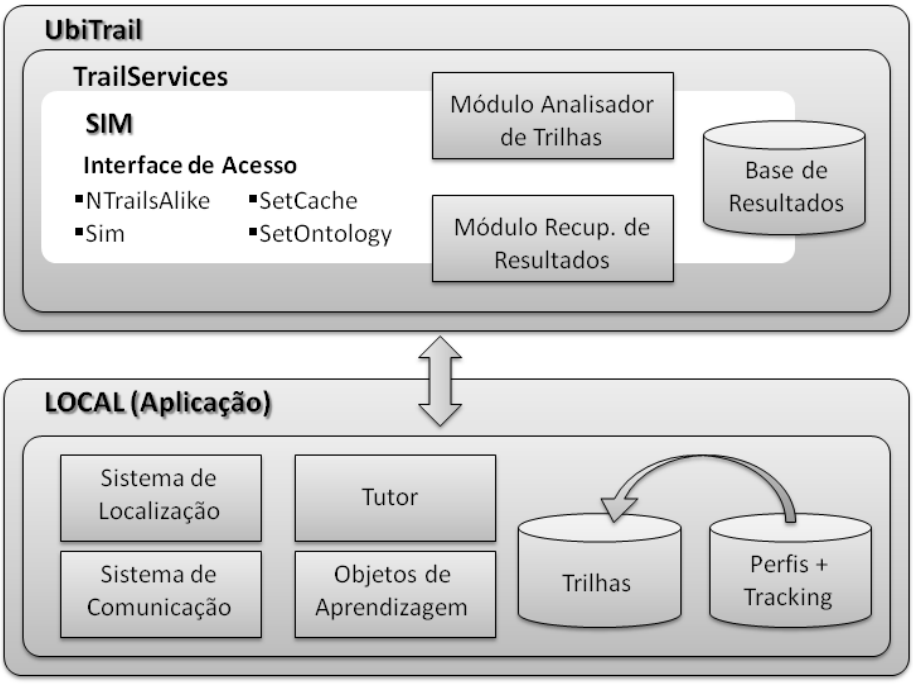

Figura 20. Visão Geral da integração entre o SIM, LOCAL e Ubitrail

A integração da arquitetura SIM com o ambiente de validação é realizada através da criação de serviços no Ubitrail (**Figura 20)**. Os componentes da arquitetura correspondem a **serviços básicos** criados na camada TrailService do Ubitrail. Estes serviços comunicam-se entre si e com o restante do sistema através das facilidades que o Ubitrail fornece/implementa. O Tutor do LOCAL acessará os serviços através do UbiTrail, utilizando os resultados para gerar recomendações e estimular a interação entre os alunos. Os serviços criados no UbiTrail para dar suporte a esta aplicação foram os seguintes:

1) Similarity: retorna o grau de similaridade entre duas trilhas. Os parâmetros são os dois fragmentos de trilha que se deseja comparar, bem como a ontologia que descreve as trilhas. É retornado o grau de similaridade encontrado entre as duas trilhas informadas, segundo a ontologia especificada.
2) NTrailsAlike: retorna uma lista de N trilhas similares à trilha informada, dentre as trilhas registradas na base de trilhas do UbiTrail. Os parâmetros são uma trilha que servirá de base para a comparação, o número de trilhas que se deseja retornar, um limiar de similaridade que deve ser respeitado, e a ontologia que descreve as trilhas. Serão retornadas as N trilhas que forem consideradas similares à trilha informada (isto é, dentro do limiar informado).

5.1 LOCAL

O LOCAL (*LOcation and Context Aware Learning*) é um ambiente orientado à criação de espaços de educação ubíqua em pequena escala. A arquitetura é formada pelos seguintes componentes (**Figura 21**): (1) Perfis de usuário, que armazenam informações específicas do aprendiz; (2) Assistente Pessoal (AP), que reside no dispositivo móvel do aprendiz; (3) Sistema de Localização, utilizado para determinar a posição física dos dispositivos móveis; (4) Repositório de Objetos de Aprendizagem, que armazena e indexa conteúdo relacionado ao processo pedagógico; (5) Sistema de Comunicação, que estabelece contato entre os diferentes subsistemas e com os aprendizes; (6) Sistema de Eventos, utilizado para agendar tarefas; (7) Tutor, um motor de análise capaz de realizar inferências com base nos dados dos perfis e do Sistema de Localização.

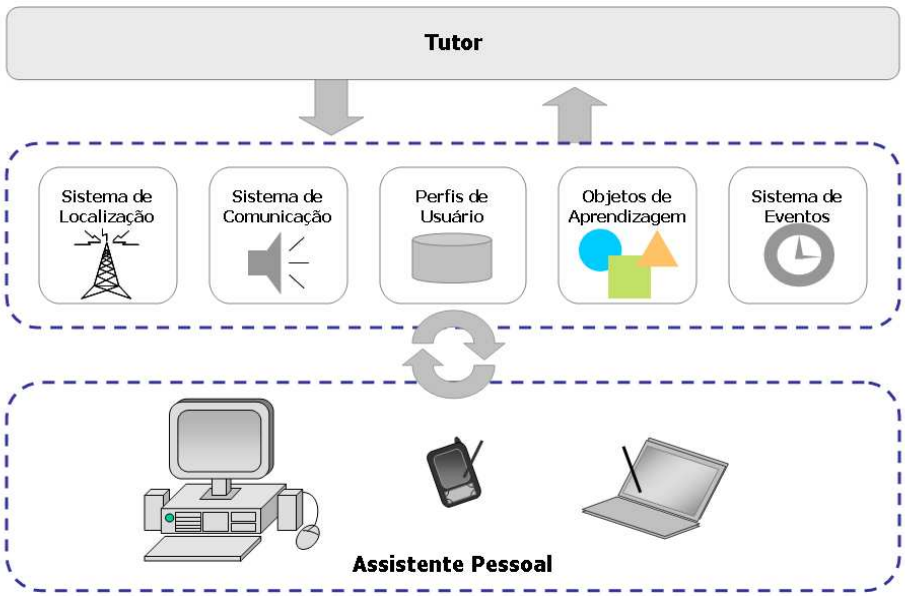

Figura 21. Arquitetura do LOCAL

Os perfis de usuários no LOCAL são baseados no padrão PAPI [58]. Estes são formados por uma parte persistente (Contato, Preferências e Interesses), que acompanham o aprendiz no seu dispositivo móvel, e por uma parte móvel (Relacionamentos, Desempenho e Segurança) vinculada aos contextos nos quais o aprendiz interage. Os perfis armazenam tanto informações específicas do aprendiz como dados de customização do processo de aprendizagem. O objetivo destes perfis é permitir a exploração de oportunidades pedagógicas nos ambientes pelos quais o aluno se desloca. A **Figura 22** mostra a organização do sistema de perfis do LOCAL. Os dados de interesses dos perfis seguem as áreas de conhecimento da

ACM (*ACM Computing Classification System* [59]), sendo definidos por palavras-chave, e são relacionados a metas, como “aprender” ou “ensinar”. A seção Relacionamentos define graus de relacionamento com outros usuários do modelo (como “professor” ou “aluno”).

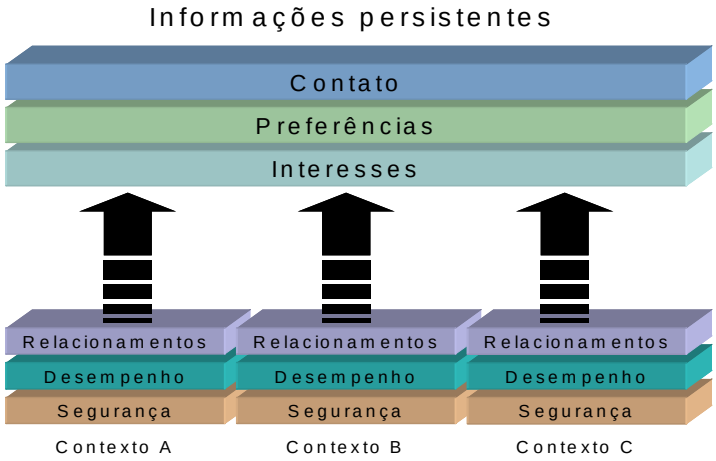

Figura 22. Sistema de perfis do LOCAL

Em relação à organização dos elementos físicos do modelo, o Assistente Pessoal (AP) é o módulo que acompanha o aprendiz. Ele é executado em um dispositivo móvel que o aprendiz carrega consigo. Ele dá suporte à autenticação na rede (segurança, recebimento de notificações da parte do Tutor, e localização precisa e constante. O Sistema de Localização do LOCAL permite a determinação da posição física de um aprendiz em um espaço de aprendizagem. O sistema vincula informações de localização física com nomes simbólicos (contextos), permitindo o mapeamento dos deslocamentos de um dispositivo móvel. Todas as mudanças de contexto são registradas (*tracking*), e servem para personalizar o processo de aprendizagem, juntamente com os dados dos perfis de usuário.

Com base nas informações dos perfis e de localização, o LOCAL permite disponibilizar objetos de aprendizagem personalizados, adequados ao contexto e grau de familiaridade dos aprendizes com determinado assunto. O sistema de localização informa a localização do aprendiz (contexto) para o tutor , que usa essas informação, juntamente com o perfil do aprendiz, para encaminhar objetos de aprendizagem ou sugerir a interação com outros aprendizes interessados nos mesmos assuntos. Isso é realizado através de um sistema de comunicação, gerenciado automaticamente pelo Tutor. As mensagens enviadas podem ser controladas em termos de localização (em uma sala específica), indivíduo (um aprendiz específico), ou ambos. Este sistema também suporta a criação de eventos, onde notificações

são encaminhadas em um momento e ambiente apropriados, relacionados a uma atividade pedagógica pré-determinada. A **Figura 23** e a **Figura 24** demonstram, respectivamente, o gerenciamento de objetos de aprendizagem e o sistema de comunicação do LOCAL.

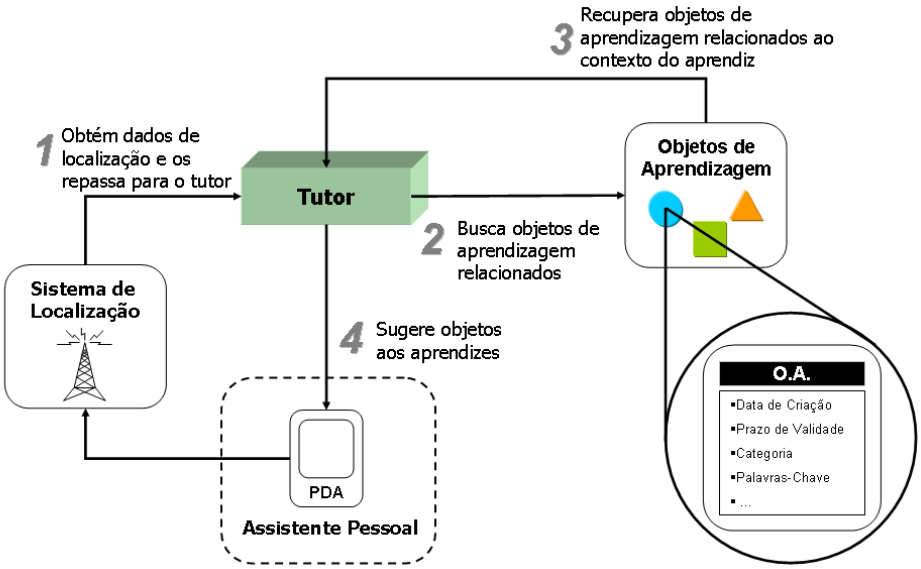

Figura 23. Objetos de aprendizagem

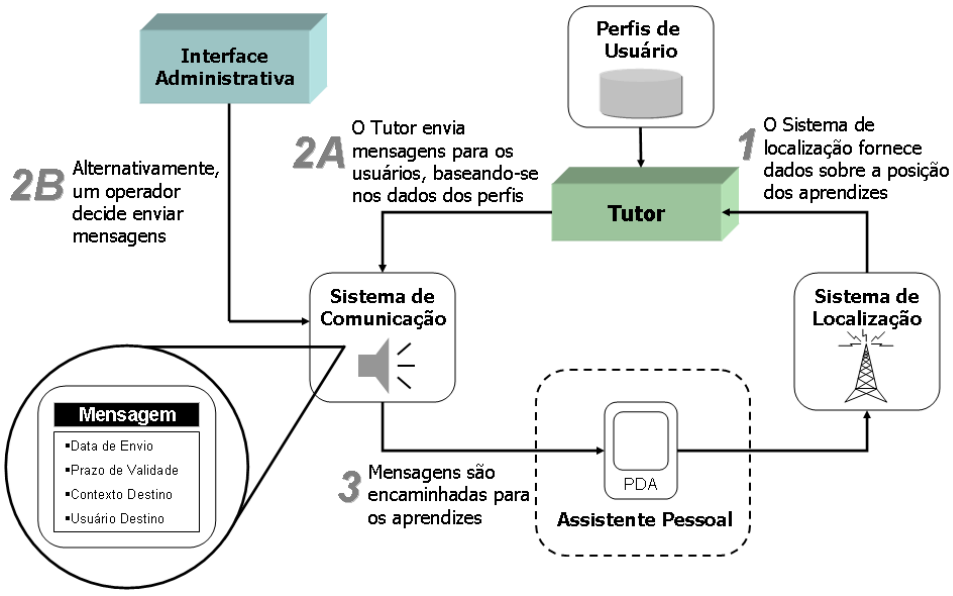

Figura 24. Sistema de comunicação

O Tutor utiliza os perfis e as informações de localização para inferência de oportunidades pedagógicas, sob a forma de recomendação de material pedagógico relevante e estímulo à interação entre aprendizes, seja no caso de interesses similares ou complementares. Um exemplo desta aplicação é a formação de grupos de estudo com base em afinidades nos perfis. A **Figura 25** exemplifica a atuação do Tutor estimulando a interação entre aprendizes através de similaridade. O Tutor descobre quem está em um contexto (passo 1), detecta um

interesse em comum entre os aprendizes (passo 2) e envia mensagens para ambos estimulando a interação (passo 3).

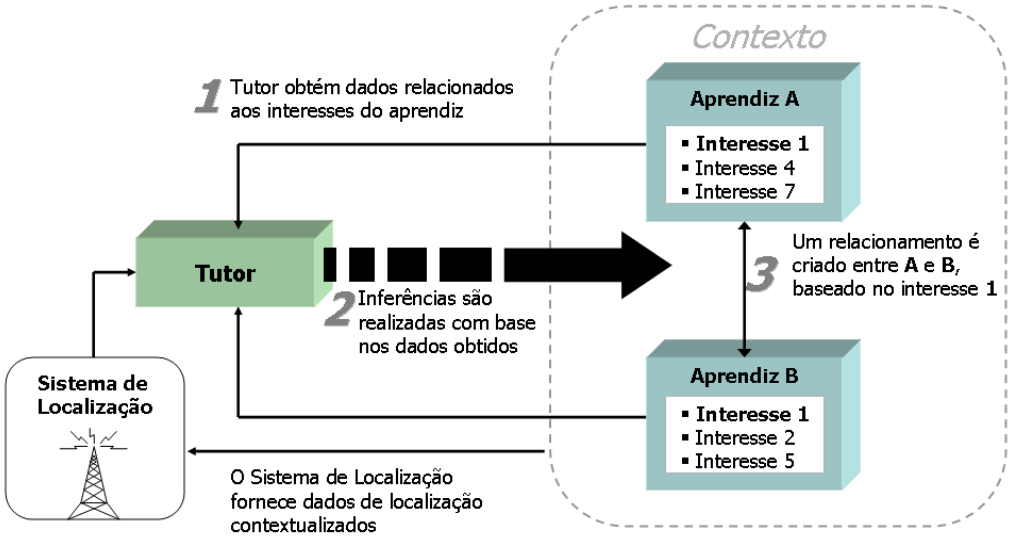

Figura 25. Funcionamento do tutor

5.2 UBITRAIL

O objetivo principal do Ubitrail é o gerenciamento de trilhas de entidades. Entidade é tudo aquilo que interage com recursos computacionais em um ambiente, ou então recursos dentro deste ambiente. Por exemplo, uma pessoa acessando determinado recurso pedagógico através de um dispositivo é uma entidade, assim como também os próprios recursos. De acordo com o Ubitrail, as aplicações implementam uma estrutura interna chamada pTrail (*piece of trail*), que armazena as informações relacionadas a um determinado ponto de trilha. Os dados da ptrail são representados de acordo com uma ontologia específica, que define os atributos e valores permitidos em cada ponto de trilha. Estes dados podem descrever entidades, recursos, eventos e locais. As trilhas propriamente ditas são compostas por sequências de registros da ptrail. Estes registros, por sua vez, são compostos por atributos organizados em três categorias: Identification, Content e Properties. A **Figura 27** demonstra a composição de trilhas no UbiTrail, processo este denominado TrailPoint. O TrailPoint ocorre a cada instante em que uma aplicação-cliente detecte um evento. Os eventos podem ser baseados no deslocamento ou interação entre as entidades, ou quando uma entidade acessa um recurso disponível no ambiente. É durante este processo que as informações componentes da ptrail passam a fazer parte da trilha de uma determinada entidade.

A **Figura 26** apresenta uma visão geral do modelo proposto no Ubitrail. O modelo segue uma arquitetura clássica cliente-servidor, com dois componentes básicos: O UbiTrailServer gerencia as trilhas e provê diversos serviços para as aplicações, enquanto

que o UbiTrailClient realiza a comunicação com o servidor e entre dispositivos móveis onde é executado.

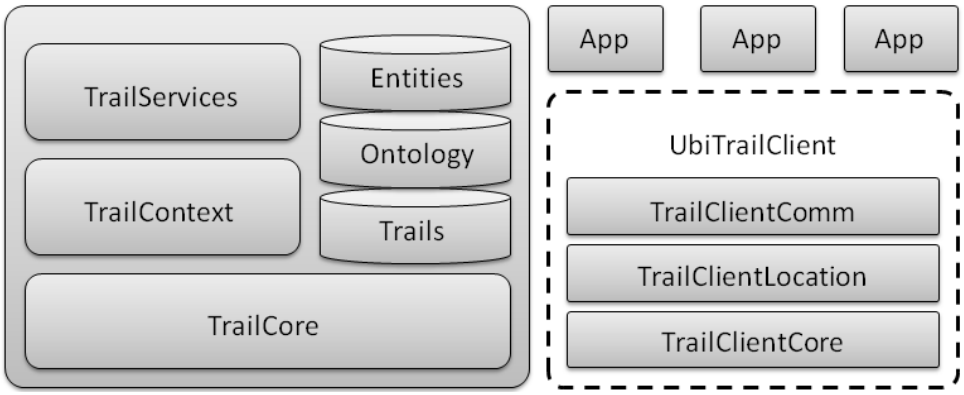

Figura 26. Modelo geral do Ubitrail (conforme [21])

A **Figura 27** exemplifica a criação de uma trilha no UbiTrail. Pontos de trilha são gerados conforme o usuário (entidade) se desloca através das regiões geográficas que compõem o domínio da aplicação. As estrelas sinalizam a ocorrência de eventos.

O UbiTrail possui uma ontologia própria, baseada no UbisWorld [60]. No entanto, para fins de validação, o SIM empregou uma outra ontologia, que retrata mais fielmente a organização dos perfis do LOCAL sob a forma de trilhas. A ontologia foi criada de modo a validar o modelo, padronizando as informações das trilhas de uma forma semelhante aos perfis de usuário do LOCAL. A **Figura 32** exibe a ontologia empregada. Esta ontologia correspondente ao documento RDF exibido na**Figura 31**.

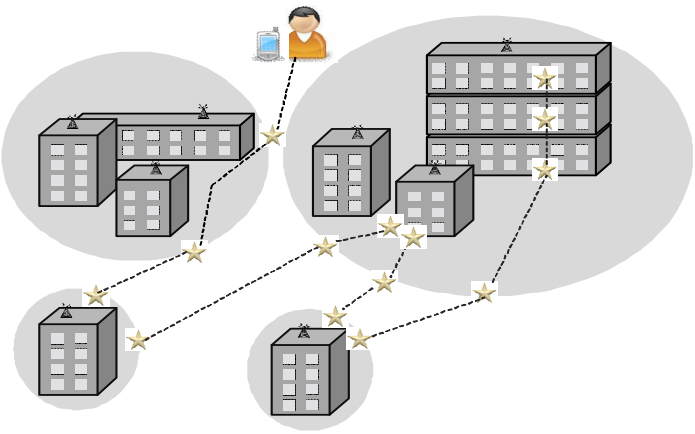

Figura 27. Composição de trilhas no Ubitrail

O UbiTrail assume a disponibilidade de um provedor de localização que ofereça informações de localização confiáveis. Este provedor pode empregar diversas métodos de localização (GPS/A-GPS, triangulação WiFi, bluetooth), compondo uma estratégia híbrida. O Ubitrail acessa periodicamente os provedor de localização, de modo a atualizar a sua visão

interna do contexto. Os contextos podem ser classificados como físicos (mensuráveis via sensores, como microfones, câmeras e termômetros) e lógicos (detectáveis através de eventos no ambiente, como interação entre usuários).

O UbiTrailServer foi projetado de maneria a aceitar dados de fontes genéricas. A primeira delas é o próprio UbiTrailClient, que fornece informações sobre as entidades, advindas do atributo Extension de cada ptrail. A segunda é o provedor de localização, que identifica as características do ambiente através de sensores espalhados neste. Com estes dados, o Ubitrail compõe as trilhas das entidades. O servidor onde o UbiTrail é executado possui três bases de dados (Entidades, Ontologia e Trillhas) que armazenam as informações pertinentes ao processo.

A parte do modelo responsável pela gerência de trilhas e acesso de terceiros ao sistema é um módulo chamado TrailServices (**Figura 28**). Este módulo fornece meios básicos através dos quais as aplicações interagem com o UbiTrail, de acordo com suas necessidades específicas. A **Figura 28** exibe uma configuraçãode TrailServices orientada a uma aplicação pedagógica que envolve adaptação de conteúdo e recomendação de material didático.

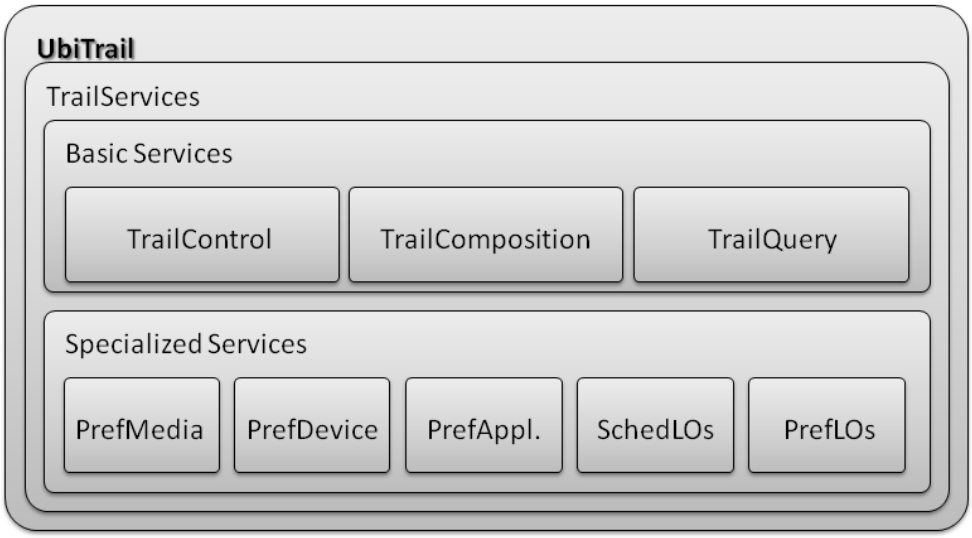

Figura 28. Detalhamento da camada TrailServices do UbiTrail

Os grupos de serviços elementares são chamados **Basic Services**. Estes serviços possibilitam a realização de tarefas simples de gerenciamento e consulta de trilhas. Os grupos de serviços básicos são: a) TrailControl (configuração, autenticação e gerência de aplicações de terceiros); b) TrailComposition (serviços empregados na composição de trilhas, como o registro de entidades, notificações de início ou término do período de composição de uma trilha e envio dos ptrails durante o processo de registro de trilha, e; c) TrailQuery (consultas sobre os dados que compõem as trilhas). O grupo TrailQuery fornece informações básicas

para as aplicações, como o conteúdo das trilhas e dados temporais. Além dos serviços básicos, o UbiTrail oferece a possibilidade das aplicações customizarem o modelo, oferecendo serviços especializados (*specialized services*), que podem ser compostos por serviços básicos (similar à noção *web* 2.0 de *mashups*) ou serem independentes. A **Tabela 7** descreve alguns serviços especializados criados de modo a estender as funcionalidades do UbiTrail.

Tabela 7. Descrição dos serviços especializados no Ubitrail (conforme [21])

Nome	Descrição
PreferredMedia	Retorna uma lista contendo os tipos de midia preferidos pelo utilizador em um contexto (por exemplo, texto, vídeo e áudio).
PreferredDevices	Retorna uma lista com os dispositivos preferidos pelo utilizador em um contexto.
PreferredApplications	O mesmo, para aplicações.
SchedulingLOs	Retorna uma lista contendo sugestões de agendamento para o acesso a recursos pedagógicos por parte do utilizador.
PreferredLOs	Retorna uma lista com os recursos pedagógicos mais acessados pelo utilizador em um contexto.

A comunicação se dá através do componente TrailContext. Toda vez que uma ptrail é recebida pelo componente TrailServices, seu conteúdo é analisado em busca de associações com informações de um contexto físico. Caso exista, o componente TrailServices aciona o componente TrailContext para obter os dados necessários.

O cliente do Ubitrail possui um componente local chamado TrailCore, que centraliza funcionalidades comuns às aplicações. Este contém métodos que possibilitam a organização, gerência local e consulta dos dados das trilhas. Estes métodos estão divididos em três famílias: a) TrailManager (consulta às ontologias e organização local dos dados das trilhas); b) TrailRegister (registro de entidades e acesso às informações destas), e; c) TrailRepository (interface de acesso à base de trilhas). O componente UbiTrailClient é executado em dispositivos móveis, e desempenha basicamente três tarefas: a) Comunicação com o UbiTrailServer; b) Obtenção de localização, e; c) Detecção de eventos realizados pelas entidades nos ambientes controlados pelo UbiTrail. Estas três tarefas estão divididas em três componentes do UbiTrailClient: a) TrailClientLocation (obtém informações de localização); b) TrailClientComm (realiza comunicação com o servidor), e; c) TrailClientCore (controla as atualizações dos ptrails no servidor). As entidades que compõem o sistema não interagem diretamente com o UbiTrailClient. Estas interagem com as aplicações, que por sua vez interagem com o UbiTrailClient, que atualiza o servidor com os eventos gerados nos contextos.

5.3 Implementação

Em termos técnicos, a interface de consulta foi implementada como um conjunto de métodos em um *webservice* que reside na camada *TrailServices* do Ubitrail. Este *webservice* tem o objetivo de disponibilizar publicamente os serviços de similaridade para as aplicações. Estas podem obter uma lista dos serviços disponíveis via WSDL (*Web Service Description Language*). A comunicação se dá via protocolo SOAP (*Service Oriented Architecture Protocol*) [61]. O Módulo Analisador é ativado por estes métodos de modo a produzir os resultados desejados nas consultas. A análise efetuada pelo Módulo Analisador segue os algoritmos apresentados na **Seção 4.3**.

A base de dados de resultados de comparação foi criada no *Microsoft SQL Server*. O Módulo de Recuperação de Resultados acessa esta base para fornecer os resultados das comparações às aplicações. O cache de respostas foi implementado como um *DataSet* (conjunto de dados) residente no espaço de memória do Módulo Recuperador de Resultados. Isso faz com que este não precise ir até o banco de dados para retornar todos os resultados de todas as comparações. Os resultados residentes neste cache de objetos interno obedecem o critério de cache configurado por cada aplicação através do TrailService **SetCacheStrategy**.

As trilhas utilizadas no teste foram criadas a partir da extração de dados de *tracking* obtidos em um período de 6 meses de utilização do LOCAL na Unisinos. Originalmente, este *tracking* continha os seguintes dados:

a) UserID - Identificador do usuário/aprendiz;
b) Context - Identificador do contexto físico (localização)
c) Access - tipo de operação efetuada dentro do contexto;
    a. Entrada em contexto.
    b. Saída de contexto.
    c. Acesso a objeto de aprendizagem dentro de um contexto.
d) Datetime - Data e hora da criação do tracking (formato yyyy-mm-dd hh:mm:ss)

Além disso, os dados extraídos continham tabelas que descreviam os contextos textualmente, bem como os objetos de aprendizagem acessados de dentro de cada contexto. A partir destes dados, pode-se fazer algumas considerações. Em primeiro lugar, a partir dos identificadores dos contextos, seus dados podem ser obtidos. Com um único campo datetime pode-se estimar a data de entrada e de saída dos contextos (convencionando-se que a data de entrada no contexto N+1 é a hora de saída no contexto N). A partir dos dados que descrevem os contextos, pode-se extrapolar um conjunto de conhecimentos relacionados (por exemplo,

no contexto “aula de Java” pode-se admitir que o conhecimento/interesse “Java” possui relevância). Estes foram mapeados para elementos **interesse** na ontologia apresentada nas **Figura 31 e Figura** 32. Além disso, o acesso a objetos de aprendizagem também foi mapeado, e está relacionado aos interesses dos aprendizes. Isso faz com que seja possível recriar as trilhas dos aprendizes, servindo de ponto de partida para os testes. O quadro da **Figura 29** mostra a equivalência de informações dos campos presentes no *tracking* do LOCAL em relação às trilhas empregadas nos testes.

Tracking	User	Context	LearningObject	Event
UserID	ID	ID	ID	ID
ContextID	Name	Name	Name	Type
EventType	Bio	Description	Subject	
ObjectID	Interests	Location	Keywords	

Figura 29. Mapeamento dos dados de tracking do LOCAL para as trilhas

Para a descrição das ontologias, usou-se a especificação RDF (*Resource Description Framework*) [62]. Esta especificação foi usada por diversos motivos:

1) É padronizado e recomendado pelo W3C desde 2004;
2) Pode ser serializada em XML, tornando muito fácil sua manipulação;
3) É madura, tem ferramentas relativamente poderosas;
4) Possui toolkit integrado com a linguagem/framework escolhida para a realização do trabalho.

Embora este não seja o foco do trabalho, foi necessário um esforço de adaptação do UbiTrail a fim de suportar a avaliação através da aplicação de teste. Neste caso, o LOCAL sofreu duas adaptações:

a) O Assistente Pessoal do LOCAL foi adaptado de modo a exibir um link para um site na rede local, denominado “Pessoas como Eu”. Este site mostra os usuários que estão no mesmo contexto do utilizador, bem como uma indicação geral de similaridade na forma de uma lista ordenada de usuários decorados com carinhas (*smilies*). O site é acessado no *Pocket Internet Explorer*;
b) O Tutor foi adaptado de modo a expor um método acessível via web, chamado **SIM_NotifyAll**. Este método recebe como parâmetro a trilha do usuário atual, e é invocado quando este entra em um contexto. Ele retorna uma lista contendo os dados público de identificação dos outros aprendizes no contexto. Esta lista é ordenada de acordo com o grau de similaridade do usuário com os outros aprendizes no contexto.

```
MemoryStore store = new MemoryStore();

Entity pessoa = new Entity("tempuri://pessoa");
Entity pesquisador = new Entity("tempuri://pesquisador");
Entity aluno = new Entity("tempuri://aluno");
Entity disciplina = new Entity("tempuri://disciplina");
Entity assunto = new Entity("tempuri://interesse");
Entity recurso = new Entity("tempuri://recurso");
Entity recursoFisico = new Entity("tempuri://recurso-f");
Entity recursoLogico = new Entity("tempuri://recurso-l");
Entity contexto = new Entity("tempuri://contexto");
Entity local = new Entity("tempuri://local");

Entity is_one_of = "tempuri://is_one_of";
Entity ministra = "tempuri://ministra";
Entity é_interessado_em = "tempuri://e_interessado_em";
Entity é_abordado_em = "tempuri://é_abordado_em";
Entity cursa = "tempuri://cursa";
Entity possui = "tempuri://possui";
Entity trata_de = "tempuri://trata_de";
Entity possui_conteudo_sobre = "tempuri://possui_conteudo_sobre";
Entity localiza_se_em = "tempuri://localiza_se_em";

store.Add(new Statement(aluno, is_one_of, pessoa));
store.Add(new Statement(pesquisador, is_one_of, pessoa));
store.Add(new Statement(pessoa,é_interessado_em,assunto));
store.Add(new Statement(pessoa,possui,recurso));
store.Add(new Statement(aluno, cursa, disciplina));
store.Add(new Statement(pesquisador, ministra, disciplina));
store.Add(new Statement(assunto, é_abordado_em, disciplina));
store.Add(new Statement(recurso, possui_conteudo_sobre, assunto));
store.Add(new Statement(recurso, is_one_of, recursoFisico));
store.Add(new Statement(recurso, is_one_of, recursoLogico));
store.Add(new Statement(recurso, trata_de, assunto));
store.Add(new Statement(recursoFisico, localiza_se_em, contexto));
store.Add(new Statement(contexto, localiza_se_em, contexto));
store.Add(new Statement(pessoa, localiza_se_em, contexto));
```

Figura 30. Criação de ontologia usando a biblioteca SemWeb

As adaptações no LOCAL em si foram desenvolvidas em C#, já que praticamente toda a infraestrutura foi criada em cima do *framework* .NET, incluindo o Assistente Pessoal. Por conveniência, o site “Pessoas como Eu” foi implementado em PHP. Ele funciona lendo um arquivo texto gerado pelo Tutor, quando este invoca o serviço **NTrailAlike** do UbiTrail para a determinação da similaridade entre as trilhas dos usuários. De posse das trilhas, este retira apenas os dados de identificação, retornando uma lista simples para o AP, que a exibe para o usuário. Se o nome de um usuário for “clicável” no site “Pessoas como Eu” (ou seja, o usuário permitiu o acesso a seus dados pessoais), pode-se visualizar o grau de similaridade entre os dois usuários, bem como os dados que compõem a lista de interesses das trilhas, lado a lado. Para controlar esta funcionalidade, foi criado um parâmetro de configuração no AP, que regula a privacidade das informações das trilhas para outros usuários (isto é, se os dados que compõem uma trilha de um usuário qualquer devem estar visíveis para os outros usuários).

```
<?xml version="1.0"?>
<rdf:RDF xmlns:ex="http://example.org/"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:ns="tempuri://">
  <rdf:Description rdf:about="tempuri://aluno">
    <ns:is_one_of>
      <rdf:Description rdf:about="tempuri://pessoa">
        <ns:e_interessado_em>
          <rdf:Description rdf:about="tempuri://interesse">
            <ns:é_abordado_em rdf:resource="tempuri://disciplina" />
          </rdf:Description>
        </ns:e_interessado_em>
        <ns:possui>
          <rdf:Description rdf:about="tempuri://recurso">
            <ns:possui_conteudo_sobre rdf:resource="tempuri://interesse" />
            <ns:is_one_of>
              <rdf:Description rdf:about="tempuri://recurso-f">
                <ns:localiza_se_em>
                  <rdf:Description rdf:about="tempuri://contexto">
                    <ns:localiza_se_em rdf:resource="tempuri://contexto" />
                  </rdf:Description>
                </ns:localiza_se_em>
              </rdf:Description>
            </ns:is_one_of>
            <ns:is_one_of rdf:resource="tempuri://recurso-l" />
            <ns:trata_de rdf:resource="tempuri://interesse" />
          </rdf:Description>
        </ns:possui>
        <ns:localiza_se_em rdf:resource="tempuri://contexto" />
      </rdf:Description>
    </ns:is_one_of>
    <ns:cursa rdf:resource="tempuri://disciplina" />
  </rdf:Description>
  <rdf:Description rdf:about="tempuri://pesquisador">
    <ns:is_one_of rdf:resource="tempuri://pessoa" />
    <ns:ministra rdf:resource="tempuri://disciplina" />
  </rdf:Description>
</rdf:RDF>
```

Figura 31. Ontologia resultante do código fonte mostrado na Figura 30

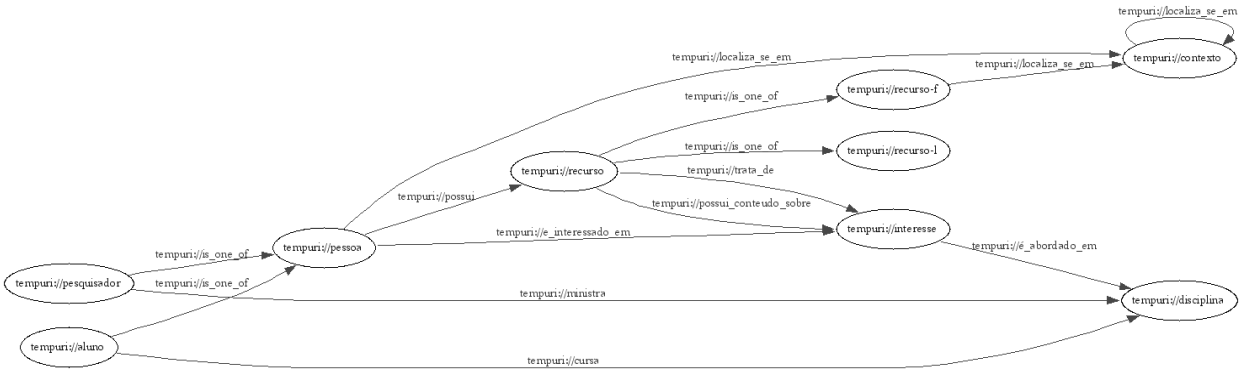

Figura 32. Representação gráfica da ontologia criada

5.4 Avaliação

De modo a avaliar o modelo, um protótipo foi implementado, integrando o SIM com o Ubitrail e o LOCAL. Os serviços da **Interface de Consulta** do SIM foram mapeados para novos serviços avançados na camada TrailService do Ubitrail:

a) **Similarity**, que retorna a similaridade entre duas trilhas informadas;
b) **NTrailsAlike**, que retorna um conjunto de trilhas similares a uma trilha informada, respeitando um limiar de similaridade desejado.
c) **SetCacheStrategy**, que configura a estratégia de cache a ser utilizada.

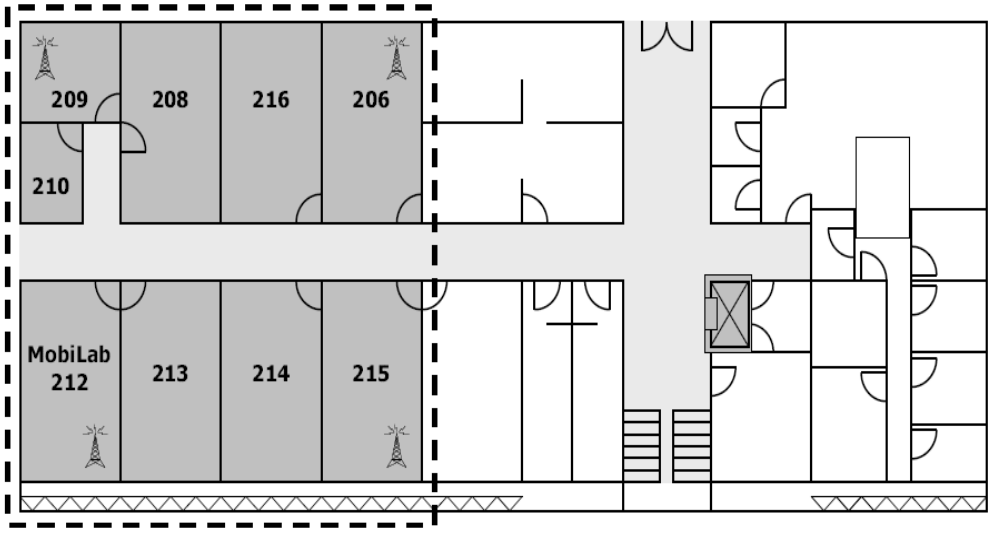

Figura 33. Cenário do teste simulado

Devido à complexidade de se configurar um ambiente real para os testes, optou-se por realizar um teste simulado. Para subsidiar os testes, foram empregadas as trilhas obtidas ao minerar a base de dados do LOCAL. Foi reproduzida uma aula simulada com a duração de

duas horas e meia, envolvendo dez participantes de um curso da área de Informática na Unisinos. O assunto da aula simulada foi “Paradigmas de Linguagens de Programação”. Os alunos portaram diversos tipos de dispositivos móveis, nos quais foram executadas instâncias do Assistente Pessoal (AP) do LOCAL. Estas instâncias do Assistente Pessoal possibilitaram a monitoração precisa e constante da localização dos alunos durante o teste. Além disso, cada acesso a recursos presentes no ambiente foi monitorado. A compilação desses dados permitiu a criação de trilhas para cada um desses alunos. Os resultados foram organizados em quatro períodos.

*O **primeiro período** ocorre antes da aula. O professor utiliza o LOCAL para agendar uma mensagem, avisando os alunos do tema da aula, bem como um link para um recurso web contendo material relacionado ao assunto. Todo o aluno que adentrar o contexto físico reservado para a realização da aula receberá a mensagem e o link contendo o material a ser acessado.*

*O **segundo período** corresponde ao início da aula, quando os alunos começam a chegar na sala. Após o login desses alunos no LOCAL, o sistema começa a coletar os pontos de trilha de cada um deles. Os alunos recebem a notificação cadastrada antes do início da aula pelo professor e, portanto, começam a acessar o material que lhes foi indicado. A partir daqui, os alunos começam se deslocam pelos contextos que compõem o ambiente de testes. Este ambiente foi mapeado por ocasião da criação do LOCAL, compreendendo 8 salas no prédio 6B da Unisinos (***Figura 33***). Nestas salas, os alunos interagem em grupo, trocam mensagens e acessam material pedagógico relacionado às disciplinas do curso. Estas interações significam a criação de eventos nas trilhas de cada aluno. Os eventos de trilha gerados correspondem a eventos reais (no ambiente simulado), como acesso a material pedagógico nos contextos, ou acessos a temas em comuns em um período de tempo.*

*No **terceiro período** o professor solicita ao Tutor a do LOCAL criação de grupos como condição para a realização de um trabalho em sala de aula. Quando um aluno ingressava em um contexto qualquer, o Assistente Pessoal do LOCAL notificava o Tutor deste novo ingresso. Normalmente, o tutor cruza os dados de perfis estáticos de usuários, localmente, de modo a procurar correspondências e identificar oportunidades pedagógicas (parcerias e recomendações de material didático). No entanto, esta instância do LOCAL foi adaptada para a manipulação de trilhas, através da integração com o UbiTrail. Mais precisamente, um serviço de extensão, na camada **SpecializedServices**, torna acessível a funcionalidade de comparação de similaridade de trilhas oportunizada pelo SIM. Dessa*

forma, ao detectar o ingresso de um aluno no contexto, o Tutor selecionou a trilha do aluno e chamou o serviço avançado ***NTrailsAlike*** *do UbiTrail (que retorna um grupo de tamanho configurável contendo as trilhas que mais se parecem com a trilha informada). Este serviço então comunicou-se com o SIM, que por sua vez, buscou trilhas similares àquela que foi informada. Ao obter os resultados, estes foram encaminhados novamente ao UbiTrail, que se encarregou de disponibilizá-los ao Tutor do LOCAL. Os grupos são organizados de acordo com o retorno do Tutor do LOCAL. Neste caso, foram formados três grupos, um deles (grupo A) formado de quatro alunos interessados em Java. O outro (grupo B) formado por dois alunos com interesse em C# e o terceiro (grupo C) com apenas um aluno interessado em C++.*

No ***quarto período*** *o trabalho é iniciado. Cada grupo apresenta as características (vantagens e desvantagens) de cada linguagem. Chegam, ainda, dois alunos atrasados, que são autenticados pelo sistema. Ambos são alunos novos na turma daquela disciplina, e recebem a mensagem sobre o conteúdo da aula do dia, sendo informados sobre quais grupos devem procurar. Um deles é encaminhado para o grupo A e o outro para o grupo C. A aula continua com a chegada destes novos alunos. Um deles, de posse de uma instância do Assistente Pessoal do LOCAL, resolve acessar o recurso "Pessoas Como Eu", disponível dentro do Assistente Pessoal, em busca de informações sobre os colegas. Este recurso exibe os alunos cujas trilhas mais corresponderam àquela do aluno recém chegado. O recurso mostra apenas informações que podem ser diretamente mapeadas para as partes públicas dos dados de cada um dos alunos caso estes estivessem dispostos como perfis (nome, email, foto) no LOCAL. A lista é ordenada de acordo com o grau de similaridade detectado pelo Tutor entre o aluno recém chegado e os outros alunos no contexto. Para aqueles alunos que assim desejam, a lista exibida no recurso "Pessoas Como Eu" permite o acesso à lista particular que compõe seus interesses, conforme os dados dispostos em cada trilha individual, bem como o grau de similaridade entre os alunos (dado pelo serviço avançado Sim do UbiTrail, que retorna o grau de similaridade entre duas trilhas informadas). O acesso é controlado através de um mecanismo de privacidade composto por dois itens: um parâmetro de configuração no Assistente Pessoal, e uma flag no Tutor que indica se o acesso deve ser disponibilizado publicamente ou não. Ao receber os dados delineando a similaridade entre as suas trilhas e as dos demais colegas, os alunos buscam, espontaneamente, a interação com os colegas. A interação, de um modo geral, espelha os grupos mais similares entre si, conforme a análise do Tutor encaminhada ao SIM através do UbiTrail.*

De um modo geral, a integração exibida através do teste simulado mostrou que é possível montar aplicações que se beneficiem da análise de dados de trilhas. Além disso, demonstrou o aspecto “plugável” da solução criada, ao detalhar a maneira como esta foi integrada ao LOCAL e ao UbiTrail, a troca de dados e de resultados, bem como a possibilidade de reutilização dos componentes de *software* que compõem a solução. Neste caso, o SIM foi implementado como um componente dentro do Ubitrail, o UbiTrail ofereceu o armazenamento e a infraestrutura de serviço (*Specialized Services*), enquanto o LOCAL ofereceu a infra-estrutura de acesso (Assistente Pessoal e Tutor).

Especificamente, os resultados mostram que o sistema advindo da integração entre o LOCAL/UbiTrail e o SIM pode ser usado para criar grupos de alunos com trilhas similares, potencializando dessa forma a interação entre os alunos. Embora um ambiente simulado não possibilite receber feedback relacionado a questões de usabilidade, mesmo assim torna-se evidente a vantagens advindas da utilização de um sistema similar como ferramenta de apoio ao processo pedagógico. Através da interação com elementos contextuais, é possível extrair informações que servem de ponto de partida para aplicações inovadoras.

A aplicação apresentada na simulação foi um exemplo específico (neste caso, um ambiente de educação). No entando, percebe-se a viabilidade de implantar soluções similares em outros contextos. O **Capítulo 6** promove o fechamento do trabalho e enumera algumas aplicações possíveis de um sistema similar ao apresentado aqui, em outros contextos/ambientes.

6 CONSIDERAÇÕES FINAIS

Este capítulo traz uma análise das principais contribuições da arquitetura proposta, bem como algumas possibilidades de utilização da tecnologia aqui apresentada para criar novas aplicações.

6.1 Contribuições

O trabalho apresentou um modelo para determinação de similaridade entre trilhas que descrevem entidades genéricas. Um protótipo foi implementado, compreendendo a integração entre a arquitetura proposta e duas outras arquiteturas de suporte, uma com foco puramente educacional e outra de aplicação genérica. O modelo serviu para a criação de aplicações inovadoras, utilizando conhecimentos sobre trilhas para a obtenção de resultados que possibilitem inferir novas informações a partir de um conjunto de trilhas iniciais, e respondendo às questões de pesquisa levantadas na **Seção 1.2.**

6.2 Aplicações futuras

A tecnologia estudada tornam possível vislumbrar uma ampla gama de aplicações que podem ser beneficiadas pela arquitetura criada. Seguem alguns exemplos:

a) **Formar casais em aplicações de *"match making"*, *"speed dating"* ou similares** – este exemplo seria semelhante à aplicação de cunho pedagógico que serviu como elemento de teste nessa proposta. Neste caso, a aplicação procuraria encontrar aquela trilha mais semelhante à de um candidato qualquer, dentre as pessoas do sexo oposto. Isso exigiria uma maneira de passar parâmetros arbitrários (que não são trilhas) para a Interface de Acesso . Esta busca seria feita no conjunto de dados da empresa que promove os encontros, desde que este estivesse disposto sob a forma de trilhas;

b) **Encontrar veículos fora de rota** – dada uma base de trilhas descrevendo rotas de veículos, basta detectar o nível de similaridade entre as trilhas que se deseja analisar e uma trilha que servirá como "medida de ouro" para aquela rota particular. Esta pode ser obtida manualmente, ou então realizando-se a "média" entre as trilhas obtidas por todos os carros que fazem aquela rota. Neste caso, pode-se obter o desvio-padrão do conjunto de carros em relação à não-conformidade com a rota, e qualquer veículo que esteja fora desse desvio-padrão estará fora da rota;

c) **Detectar engarrafamentos** – com a mesma base de trilhas de veículos utilizada no item anterior, pode-se detectar anomalias no trânsito. Para isso, basta adicionar a dimensão do tempo que se leva entre dois pontos que configuram uma parte da rota. Se o tempo de deslocamento entre dois pontos de uma rota for maior do que o desvio padrão, pode haver um engarrafamento. Neste caso, as irregularidades detectadas no item "b" podem ser usadas a favor dos motoristas: uma aplicação gráfica que mostre visualmente as rotas alternativas para o deslocamento;

d) **Prever ocorrências futuras em uma trilha** – dado um conjunto suficientemente grande de trilhas (onde o "suficientemente grande" varia de aplicação para aplicação), é possível detectar aglomerações de dados em torno de temas ou de datas. Essas aglomerações representam tendências. De posse dos dados que representam essas tendências uma aplicação poderia dizer, dentre um conjunto de eventos, qual deles tem maior chance de ocorrer como próximo passo de uma trilha. Isso tem aplicações que vão desde comerciais (disposição das lojas em um shopping) até educacionais (se um aluno está desviando da trilha, merece atenção especial). Outra aplicação seria determinar se é seguro tomar determinado passo em uma trilha (por exemplo, abrir um negócio em determinada região). Neste caso, deve haver um artifício pelo qual se possa informar ao sistema quais são os fins (*outcomes*) desejáveis (faturamente aumentando a cada ano) e os indesejáveis (por exemplo, faturamento diminuindo a cada ano, ou então o evento *"decretar falência"*);

e) **Estimar a probabilidade de uma determinada ocorrência futura** – observando a base de trilhas, pode-se estimar a probabilidade de ocorrências futuras; neste caso, o usuário informaria uma trilha, bem como um fim cuja probabilidade de ocorrência se deseja detectar. Neste caso, o sistema devolveria a probabilidade de ocorrência do evento.

6.3 Trabalhos Futuros

Como trabalhos futuros, pode-se prever:

a) **Aplicações testadas em ambientes reais –** embora a aplicação apresentada seja representativa da arquitetura e traga uma contribuição razoável, esta foi desenvolvida em um ambiente simulado. Espera-se no futuro testar a instância gerada em um ambiente de aplicações reais;

b) **Escala da aplicação de exemplo –** a aplicação de exemplo foi testada em um ambiente de pequena escala. A arquitetura contempla o conceito de escalabilidade e fornece um mecanismo de controle de escalabilidade, no entanto este aspecto não foi diretamente testado. Futuramente, deve-se retornar a esse aspecto e retrabalhar os testes de modo a incluí-lo.

c) **Representatividade das aplicações genéricas –** além do caráter simulado do ambiente de validação, a aplicação constituiu um caso bastante específico. A arquitetura é genérica, e serve para muito mais cenários do que aquele na qual foi testada. Assim sendo, nos próximos trabalhos procura-se criar mais exemplos em outros ambientes, de modo a reforçar o caráter genérico da solução criada.

7 REFERÊNCIAS

[1] Mahadev Satyanarayanan, Paramvir Bahl, Ramon Caceres, Nigel Davies, "The Case for VM-Based Cloudlets in Mobile Computing," IEEE Pervasive Computing, pp. 14-23, October-December, 2009.

[2] Almudena Diaz, Pedro Merino, F. Javier Rivas, "Mobile Application Profiling for Connected Mobile Devices," IEEE Pervasive Computing, pp. 54-61, January-March, 2009.

[3] Hightower, J.; Borriella, G. Location systems for ubiquitous computing. IEEE Computer, 34(8):57–66, 2001.

[4] Practical Lessons from Place Lab. Jeffrey Hightower, Anthony LaMarca and Ian Smith. IEEE Pervasive Computing, vol. 5, no. 3, 2006.

[5] Driver C. (2007) An Application Framework for Mobile, Context-Aware Trails. PhD thesis, University of Dublin.

[6] Levene M., Peterson D. (2002) Trail Record and Ampliative Learning. Research Report BBKCS-02-01, School of Computer Science and Information Systems, University of London

[7] Weiser, M. The computer for the 21st century. SIGMOBILE Mob. Comput. Commun. Rev., ACM Press, New York, NY, USA, v. 3, n. 3, p. 3–11, July, 1999.

[8] Satyanarayanan, M. Pervasive computing: vision and challenges. Personal Communications, IEEE [see also IEEE Wireless Communications], v. 8, n. 4, p. 10–17, 2001.

[9] Vaughan-Nichols S. J. (2009) Will Mobile Computing's Future Be Location, Location, and Location? Computer, 42:14-17

[10] Anind Dey, Jeffrey Hightower, Eyal de Lara, Nigel Davies, "Location-Based Services," IEEE Pervasive Computing, Vol. 9, 01, pp. 11-12, January-March, 2010.

[11] Global Locate Inc. A-GPS Technology. http://www.globallocate.com/A-GPS/A-GPS_Frameset.htm, June 2007.

[12] Qualcomm Incorporated. gpsOne Technology. http://www.cdmatech.com/solutions/products/gpsone.jsp, June 2007.

[13] Federal Communications Comission (FCC). Enhanced 911 - Wireless Services. http://www.fcc.gov/911/enhanced, June 2006

[14] H. Rubinsztejn et al. Support for Context-Aware Collaboration. In I International Workshop on Mobility Aware Technologies and Applications (MATA), LNCS n. 3284, p.37-47, 2004.

[15] SILVA, Jader ; ROSA, João H. ; BARBOSA, Jorge L. V. ; BARBOSA, Débora N. F. ; PALAZZO, Luiz Antônio M. . Distribuição de Conteúdo em Ambientes Cientes de Trilhas. In: XV Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia), 2009, Fortaleza. Anais do XV WebMedia. Porto Alegre : SBC, 2009. p. 115-122.

[16] Lexical Similarity. http://en.wikipedia.org/wiki/Lexical_similarity. Último acesso em novembro de 2009.

[17] Semantic Similarity. http://en.wikipedia.org/wiki/Semantic_similarity Último acesso em novembro de 2009.

[18] R. Thiagarajan, G. Manjunath, M. Stumptner, "Computing Semantic Similarity Using Ontologies", *HP Labs Technical Report HPL-2008-87*, Jul. 2008.

[19] Alexander Maedche, Steffen Staab. Measuring Similarity between Ontologies Export.: Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web : 13th International Conference, EKAW 2002, Siguenza, Spain, October 1-4, 2002. Proceedings (2002), 251.

[20] Erich Gams, Tobias Berka, Siegfried Reich, Jakob Haringer Strasse. The TrailTRECer Framework: A platform for trail-enabled recommender applications. In: Conference on Database and Expert Systems DEXA ’02, Aix-en-Provence, 638-647, 2002.

[21] Jader Marques da Silva. UbiTrail: Um Modelo para Gerenciamento de Trilhas Orientado a Ambientes Ubíquos. 2009. Dissertação (Mestrado em Computação Aplicada) - Universidade do Vale do Rio dos Sinos, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior. Orientador: Jorge Luis Victória Barbosa.

[22] BARBOSA, Jorge L. V. ; HAHN, Rodrigo ; BARBOSA, Débora N. F. ; GEYER, Cláudio F R . Mobile and Ubiquitous Computing in an Innovative Undergraduate Course. In: 38th ACM Technical Symposium on Computer Science Education (SIGCSE), 2007, Covington. Proceedings of the 38th ACM SIGCSE. New York : ACM Press, 2007. v. 1. p. 379-383.

[23] BARBOSA, Jorge L. V. ; HAHN, Rodrigo ; RABELLO, Solon A ; BARBOSA, Débora N. F. . LOCAL: a Model Geared Towards Ubiquitous Learning. In: 39th ACM Technical Symposium on Computer Science Education (SIGCSE), 2008, Portland. Proceedings of the ACM SIGCSE 2008. New York : ACM Press, 2008. p. 432-436.

[24] FRANCO, Laerte K. ; BARBOSA, Jorge L. V. ; YAMIN, Adenauer C ; ROSA, João H. Um Modelo para Exploração de Oportunidades no Comércio Ubíquo. In: XXXV Conferência Latino Americana de Informática (CLEI), 2009, Pelotas. Anais do XXXV CLEI, 2009. p. 1-10.

[25] Smith, A. D., Hall, W., Glaser, H. and Carr, L. A. (2006) Towards Truly Ubiquitous Life Annotation. In: Memories for Life Colloquium 2006, 12th December 2006, The British Library, London, England.

[26] GRUBER, T. R. "A translation approach to portable ontology specifications". In: *Knowledge Acquisition*. 5: p.199. 1993.

[27] GRUBER, T. R., "Toward Principles for the Design of Ontologies Used for Knowledge Sharing". In: *International Journal Human-Computer Studies*, 43(5-6):907-928, 1995

[28] Apresentação prof. Virginia Brilhante. Disponível em *www.dcc.ufam.edu.br/~ontologias/slides/ontosComput.pdf*

[29] *Web Onthology Language*. Disponível em http://www.w3.org/TR/owl-ref. Último acesso em setembro de 2009.

[30] FENSEL, D., VAN HARMELEN, F., HORROCKS, I., McGUINNESS, D. L., & PATEL-SCHNEIDER, P. F. (2001). “OIL: an ontology infrastructure for the Semantic Web”. In: *Intelligent Systems*. IEEE, 16(2): 38–45.

[31] DAML+OIL. Disponível em www.daml.org. Último acesso em setembro de 2009.

[32] Site do Protégé. Disponível em protege.stanford.edu. Último acesso em setembro de 2009

[33] Site do OntoEdit. Disponível em www.ontoknowledge.org/tools/ontoedit. Último acesso em setembro de 2009.

[34] Site do WebODE. Disponível em webode.dia.fi.upm.es/WebODEWeb/index.html. Último acesso em setembro de 2009.

[35] KIFER, M. "Rules and ontologies in f-logic," 2005, pp. 22-34. [Online]. Disponível em http://dx.doi.org/10.1007/11526988_2

[36] ANTONIOU, G. & VAN HARMELLEN, FF. *A Semantic Web Primer (Cooperative Information Systems)*. The MIT Press, April 2004. [Online].

[37] Levenshtein Distance (*edit distance*). http://en.wikipedia.org/wiki/Levenshtein_distance. Último acesso em novembro de 2009.

[38] Dominik Scheder. Using a Skewed Hamming Distance to Speed Up Deterministic Local Search, 2010.

[39] Shengyue Ji , Guoliang Li , Chen Li , Jianhua Feng, Efficient interactive fuzzy keyword search, Proceedings of the 18th international conference on World wide web, April 20-24, 2009, Madrid, Spain [doi>10.1145/1526709.1526760]

[40] Cosine Similarity. http://en.wikipedia.org/wiki/Cosine_similarity. Último acesso em novembro de 2009.

[41] Julie B. Lovins. Development of a stemming algorithm. June 1968.

[42] Angel Freddy Godoy Viera e Johnny Virgil. Uma revisão dos algoritmos de radicalização em língua portuguesa. Programa de Pós-graduação em Ciência da Informação, Universidade Federal de Santa Catarina, Florianópolis, Brasil, 2007.

[43] The Porter Stemming Algorithm. Disponível em http://tartarus.org/~martin/PorterStemmer/. Último acesso em Maio de 2009.

[44] Paolo Avesani, Fausto Giunchiglia, Mikalai Yatskevich. A Large Scale Taxonomy Mapping Evaluation. In: Proceedings of ISWC, 67-81, 2005.

[45] Spreading Activation Networks. http://en.wikipedia.org/wiki/Spreading_activation. Último acesso em novembro de 2009.

[46] Bipartite Graphs. http://mathworld.wolfram.com/BipartiteGraph.html. Último acesso em novembro de 2009.

[47] Bag of Words Model. http://en.wikipedia.org/wiki/Bag_of_words_model. Último acesso em novembro de 2009.

[48] WordNet website. http://wordnet.princeton.edu/. Último acesso em novembro de 2009.

[49] Wikipedia. http://www.wikipedia.org/. Último acesso em novembro de 2009.

[50] Cognitive Science: A Multidisciplinary Journal, Vol. 29, No. 1. (2005), pp. 41-78.

[51] Dice's Coeficient. http://en.wikipedia.org/wiki/Dice%27s_coefficient. Último acesso em novembro de 2009.

[52] Millard, D. E., Moreau, L., Davis, H. C., and Reich, S. 2000. FOHM: a fundamental open hypertext model for investigating interoperability between hypertext domains. In Proceedings of the Eleventh ACM on Hypertext and Hypermedia (San Antonio, Texas,

United States, May 30 - June 03, 2000). HYPERTEXT '00. ACM, New York, NY, 93-102. DOI= http://doi.acm.org/10.1145/336296.336334

[53] Moreau, L., Zaini, N., Cruickshank, D. and De Roure, D. (2003) SoFAR: An Agent Framework for Distributed Information Management. In: Intelligent Agent Software Engineering, pp. 49-67, Idea Group Publishing. ISBN 1-59140-046-5

[54] Cross Multiplication. http://en.wikipedia.org/wiki/Cross-Multiplication. Último acesso em novembro de 2009.

[55] Priority Queues. http://en.wikipedia.org/wiki/Priority_Queue. Último acesso em novembro de 2009.

[56] Padgham, L.; Winikoff, M. Prometheus: a methodology for developing intelligent agents, Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 1, Bologna, Italy, 2002.

[57] SemWeb. http://razor.occams.info/code/semweb/. Último acesso em janeiro de 2011.

[58] PAPI - Draft standard for learning technology - public and private information (papi) for learners (papi learner), Last August May 2009 [Available at http://www.cen-ltso.net/Users/main.aspx?put=230].

[59] ACM Computing Classification System. Last Accessed August 2009 [Available at http://www.acm.org/class/1998].

[60] UbisWorld. Last Accessed August 2009 [Av real available at http://www.ubisworld.org]

[61] SOAP Protocol. http://en.wikipedia.org/wiki/SOAP. Último acesso em janeiro de 2011.

[62] Resource Description Framework. http://en.wikipedia.org/wiki/RDF. Último acesso em janeiro de 2011.