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

ANDRÉ WAGNER

UM MODELO PARA GERENCIAMENTO DE PERFIS DE ENTIDADES ATRAVÉS DE INFERÊNCIA EM TRILHAS

SÃO LEOPOLDO
2013

André Wagner

UM MODELO PARA GERENCIAMENTO DE PERFIS DE ENTIDADES ATRAVÉS DE INFERÊNCIA EM TRILHAS

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

Orientador:
Prof. Dr. Jorge Luis Victória Barbosa

São Leopoldo
2013

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO (CIP)

Wagner, André

Um Modelo para Gerenciamento de Perfis de Entidades Através de Inferência em Trilhas / André Wagner — 2013.

86 f.: il.; 30 cm.

Dissertação (mestrado) — Universidade do Vale do Rio dos Sinos, Programa de Pós-Gradudação em Computação Aplicada, São Leopoldo, 2013.

"Orientador: Prof. Dr. Jorge Luis Victória Barbosa, Unidade Acadêmica de Pesquisa e Pós-Graduação".

1. Gerenciamento de perfis. 2. Computação ubíqua. 3. Modelagem de usuários. 4. Trilhas. 5. Inferência de perfis. I. Título.

CDU 004.732

Bibliotecária responsável: Eliete Mari Doncato Brasil — CRB 10/1184

(Esta folha serve somente para guardar o lugar da verdadeira folha de aprovação, que é obtida após a defesa do trabalho. Este item é obrigatório, exceto no caso de TCCs.)

AGRADECIMENTOS

Muitas foram as pessoas que me apoiaram e estiveram ao meu lado durante a elaboração deste trabalho. Assim, expresso a vocês os meus sinceros agradecimentos.

Agradeço primeiramente à CAPES, pelo investimento financeiro no meu curso de mestrado e pelo apoio que tem prestado à educação brasileira.

Agradeço aos meus colegas de mestrado pelos bons momentos que passamos juntos, de diversão e aprendizado, e pelo desprendimento de todos em passar seu conhecimento uns aos outros.

Agradeço à Débora Nice Ferreira Barbosa pelo apoio prestado na elaboração da proposta deste trabalho, pelas trocas de ideias e revisões feitas aos meus trabalhos e artigos.

Agradeço ao meu orientador Jorge Barbosa pelo apoio demonstrado, pelas infindáveis revisões, pelas trocas de ideias, por estar sempre disponível e pronto a responder dúvidas e questionamentos, e por demonstrar um genuíno interesse no meu trabalho.

Aos meus pais, pelo seu amor e investimento que fizeram em minha vida e minha educação, promovendo as bases para que eu obtivesse o conhecimento necessário para a realização deste trabalho.

Agradeço em especial à minha esposa Morgana, a pessoa que foi mais diretamente afetada pelo meu envolvimento neste trabalho, e que esteve todo tempo ao meu lado me apoiando e me encorajando. Eu não teria conseguido sem ela.

"Não é o crítico que conta;
não é o homem que aponta como o forte tropeça,
ou como o trabalho poderia ter sido feito melhor.
O crédito pertence ao homem que está na arena,
cujo rosto está sujo de pó, suor e sangue; que luta bravamente;
que erra, que falha de novo e de novo,
pois não há esforço sem erros e sem falhas;
mas que luta para realizar grandes feitos;
que conhece os grandes entusiasmos, os grandes sacrifícios;
que dá de si mesmo por uma causa nobre;
que, na melhor das hipóteses, descobre no fim o triunfo da grande realização
e que, na pior das hipóteses, falha,
mas ao menos falha lutando com bravura,
de modo que seu lugar nunca será entre as almas frias e tímidas
que não conhecem nem vitória nem derrota."

Theodore Roosevelt

RESUMO

Um dos principais desafios de sistemas ubíquos e sensíveis a contexto é a coleta de informações relevantes sobre entidades, e o uso destas informações para compreender e prever seu comportamento. Isto permite que as aplicações adaptem-se às entidades, evitando assim uma sobrecarga de questionamentos e informações à entidade. Este trabalho apresenta o eProfile, um modelo que permite que aplicações registrem as ações de entidades em trilhas e infiram informações de perfil a partir destas trilhas, utilizando interoperabilidade semântica e assim permitindo que diferentes aplicações compartilhem informações em um perfil unificado. Foi desenvolvido um protótipo para a avaliação do modelo, o qual foi integrado com dois diferentes softwares. Foi verificado que é possível enriquecer a geração de perfis de aplicações através da integração com o modelo. As contribuições deste modelo são o uso de trilhas para extrair perfis, a geração de perfis dinâmicos, o gerenciamento de regras de inferência e modelos de entidades dinâmicos e a interoperabilidade semântica do modelo.

Palavras-chave: Gerenciamento de perfis. Computação ubíqua. Modelagem de usuários. Trilhas. Inferência de perfis.

ABSTRACT

Context-aware and ubiquitous systems have the challenge of implicitly collect relevant information about entities, and use this information to understand and predict their behaviour. This allows the applications to adapt themselves to the entities, thus avoiding to overflow them with inquires and information. The analysis of trails, the context-aware history of actions, can further improve the relevance of information. This dissertation proposes a model that allows applications to register entites' actions in trails and infer profile information from these trails, using semantic interoperability and thus allowing different applications to share information and infer a unified profile. A prototype was developed for the evaluation of the model, and it was integrated with two different softwares. It was verified that was possible to enrich the profile generation of applications through the integration with the modelo. The contributions of this model are the use of trails for extracting profiles, the generation of dynamic profiles, the capability of managing dynamic inference rules for profile generation and the semantic interoperabilty of the model.

Keywords: Profile management. Ubiquitous computing. User modeling. Trails. Profile inference.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Conceitos Gerais

Atualmente, a maioria dos usuários interage com uma grande variedade de sistemas computacionais, tanto no trabalho quanto em casa. Entre os sistemas utilizados, podem-se citar máquinas fotográficas digitais, telefones celulares, caixas eletrônicos, aparelhos de DVD, reprodutores de música portáteis e *tablets*. No entanto, muitos dos usuários não percebem a variedade e a quantidade de sistemas utilizados diariamente.

Cada vez mais, existe a tendência do aumento da interconectividade entre estes sistemas, devido ao fato da internet estar cada vez mais presente e disponível em qualquer lugar. Assim, pesquisas recentes preveem a integração de todos estes sistemas em uma federação de sistemas computacionais, que podem ser descritos como Ambientes Inteligentes (HECKMANN, 2005).

Uma visão para o conceito de ambientes inteligentes foi discutida por Weiser (1991) em seu artigo seminal, onde ele apresentou um cenário em que o computador se tornaria invisível e indistinguível de seu ambiente. Neste cenário, a interação entre humanos e computadores ocorreria não como a usual – onde o humano dedica 100% de sua atenção ao computador –, mas constituiria-se de pequenas interações rápidas, geralmente em locais e dispositivos distintos, muitas vezes utilizando a periferia da atenção do usuário (WEISER; BROWN, 1996) ou sem ele se dar conta de estar interagindo com um sistema computacional.

Para que este cenário seja possível, é necessário que os aplicativos tenham condições de considerar o contexto no qual se encontram e adaptar-se a este contexto, ou seja, é necessário que sejam "sensíveis a contexto". Embora existam várias definições de contexto, uma das mais utilizadas é a de que um contexto é qualquer informação que pode ser usada para caracterizar a situação de uma entidade, onde uma entidade pode ser uma pessoa, local ou objeto considerado relevante à interação entre o usuário e a aplicação (DEY, 2001).

Para que um aplicativo possa se adaptar a um usuário de forma eficaz, é importante que ele conheça não apenas o histórico de contextos visitados por um usuário, mas conheça quem o usuário é. Atualmente, diversos trabalhos têm buscado maneiras de coletar informações a respeito dos usuários, com o objetivo de conhecer suas preferências, possibilitando a oferta de serviços personalizados. Assim, estas aplicações organizam seu conhecimento a respeito do usuário em perfis de usuário, os quais armazenam as informações em um formato baseado em um modelo de usuário específico (VIVIANI; BENNANI; EGYED-ZSIGMOND, 2010).

Pesquisas nesta área têm indicado que não apenas é interessante conhecer o contexto do usuário no momento em que um determinado serviço está sendo oferecido, mas que o histórico de contextos visitados pode ser uma informação valiosa na adaptação da aplicação ao usuário. Este histórico de contextos recebe o nome de Trilha (SILVA et al., 2010).

1.2 Motivação

Desde que os primeiros computadores se tornaram disponíveis, tem havido uma tendência: o custo e o tamanho dos dispositivos têm diminuído, enquanto sua presença têm se tornado cada vem mais ubíqua. E com o advento da internet, cada vez mais serviços tem sido disponibilizados em qualquer lugar que o usuário esteja. Estes aspectos deram origem a uma visão onde serviços computacionais estão disponíveis a qualquer momento e em qualquer lugar, denominada de Computação Ubíqua (WEISER, 1991).

Weiser e Brown (1996) discutem as consequências futuras, caso esta tendência continue – e não há nada que demonstre que ela irá reduzir. Caso o modelo atual de interação humano-computador seja mantido, é provável que cada vez mais os usuários estejam sobrecarregados de interações computacionais e informações, causando assim detrimento na qualidade de vida destes. Por isso, os autores apresentam o conceito de Tecnologia Calma, onde a interação entre os computadores e o humano ocorrem na periferia de atenção do usuário. Neste conceito, o computador desaparece da consciência do usuário, a não ser no momento em que seja necessário, quando então chama atenção para si.

Satyanarayanan (2001) chama a atenção para o fato de que um sistema só pode se tornar invisível se o mesmo for proativo, e só pode se tornar proativo se souber a intenção do usuário. Muitas pesquisas têm sido feitas sobre a melhor forma de se obter a intenção do usuário e compreender quem o usuário é, o que ele está tentando realizar, e como o sistema pode recomendar opções a ele ou adaptar-se a ele.

Para que um sistema tenha sucesso em obter a intenção do usuário, é necessário que ele conheça o usuário, o contexto que o usuário está inserido e as ações que ele tomou anteriormente quando se encontrava no mesmo contexto ou em contextos semelhantes. As informações do usuário podem se enquadrar em um Perfil de Usuário, enquanto as informações históricas de contexto se encaixam na ideia de Trilha.

Embora existam alguns trabalhos que busquem inferir conhecimento a respeito de um usuário a partir de sua trilha, os trabalhos anteriores operam dentro de um contexto específico (como Educação (BARBOSA et al., 2011) ou Comércio (FRANCO et al., 2011)) ou trabalham com o modelo de usuário de forma estática.

Levando em consideração estas limitações, surge a oportunidade para o desenvolvimento de um modelo que permita a inferência de perfis de entidades a partir de uma trilha. Neste sentido, torna-se possível a utilização de perfis dinâmicos, isto é, de perfis que podem ser atualizados baseados em novas informações da trilha dinamicamente. Dentro deste contexto, abre-se a possibilidade da alteração dos modelos de entidade e das regras de geração de perfis durante a execução, aumentando assim o nível de dinamicidade do modelo.

Neste trabalho, o conceito de Entidade é usado no lugar do conceito clássico de Usuário. Enquanto geralmente o usuário compreende a pessoa que está utilizando o aplicativo, uma entidade refere-se a qualquer objeto (físico ou não) capaz de controlar um aplicativo. Desta forma,

uma entidade pode ser uma pessoa, um objeto móvel (como um carro) ou imóvel (como uma sala), ou até mesmo um sistema.

1.3 Objetivos

O objetivo geral do trabalho é especificar, implementar e validar um modelo de gerenciamento de perfis chamado **eProfile**, através do qual a trilha de uma entidade possa ser processada e transformada em um perfil de entidade. O modelo a ser apresentado é um modelo genérico, ou seja, um modelo que permite trabalhar com qualquer formato de trilha e gerar qualquer perfil de entidade requerido pelas aplicações.

Assim sendo, são objetivos específicos deste trabalho:

- apresentar o estado-da-arte em gerenciamento de perfis;
- especificar o eProfile;
- desenvolver um protótipo para o modelo;
- validar o modelo a partir do protótipo.

Após um estudo detalhado de outros modelos de gerenciamento de perfis, apresentado no Capítulo 3, foi identificado um conjunto de oportunidades para o desenvolvimento deste modelos. São elas:

- Geração de perfis dinâmicos;
- Uso de trilhas para extração de perfis;
- Domínio genérico;
- Regras de inferência e modelos de entidades dinâmicos;
- Interoperabilidade semântica.

1.4 Metodologia

A viabilização dos objetivos propostos por este trabalho ocorreu através de uma metodologia cujo primeiro passo foi o estudo de modelos já propostos para suporte ao gerenciamento de perfis. O objetivo deste primeiro passo foi verificar como os modelos atuais gerenciam perfis, e quais eram os desafios e limitações da área.

O segundo passo foi a especificação do eProfile. O objetivo é determinar quais módulos são necessários para o modelo e como eles irão interagir entre si e com serviços externos.

A elaboração de um cenário para aplicação do modelo foi o passo seguinte. Esta atividade teve por objetivo explorar os ganhos que trazidos pelo modelo.

Para validar o modelo proposto foi desenvolvido um protótipo. Este desenvolvimento cobriu todos os componentes que se fizeram necessários para viabilizar o cenário descrito no passo anterior.

Por fim, foi feita uma avaliação do modelo, onde foi realizada uma simulação da operação de entidades, gerando e analisando trilhas de entidades e inferindo perfis através destas trilhas. Os dados desse experimento foram medidos e tabulados a fim de se verificar o desempenho do modelo.

1.5 Organização da Dissertação

O Capítulo 2 descreve os principais conceitos referentes às áreas relacionadas a este trabalho. O Capítulo 3 descreve trabalhos relacionados na área de gerenciamento de perfis, apresentando os principais desafios da área. No Capítulo 4 é proposto o modelo do eProfile, bem como sua forma de operação. O Capítulo 5 apresenta os aspectos de implementação e avaliação do modelo. Por fim, o Capítulo 6 apresenta as considerações finais.

2 CONCEITUAÇÃO

Computação ubíqua, sensibilidade a contexto, trilhas e personalização foram as áreas consideradas relevantes para este trabalho. Este capítulo apresenta uma breve explicação sobre cada uma delas.

2.1 Computação Ubíqua

Em 1991, uma equipe do Laboratório de Ciência Computacional da *Xerox Palo Alto Research Center*, liderada por Mark Weiser, identificou um conjunto de tendências relacionadas à computação (WEISER, 1991):

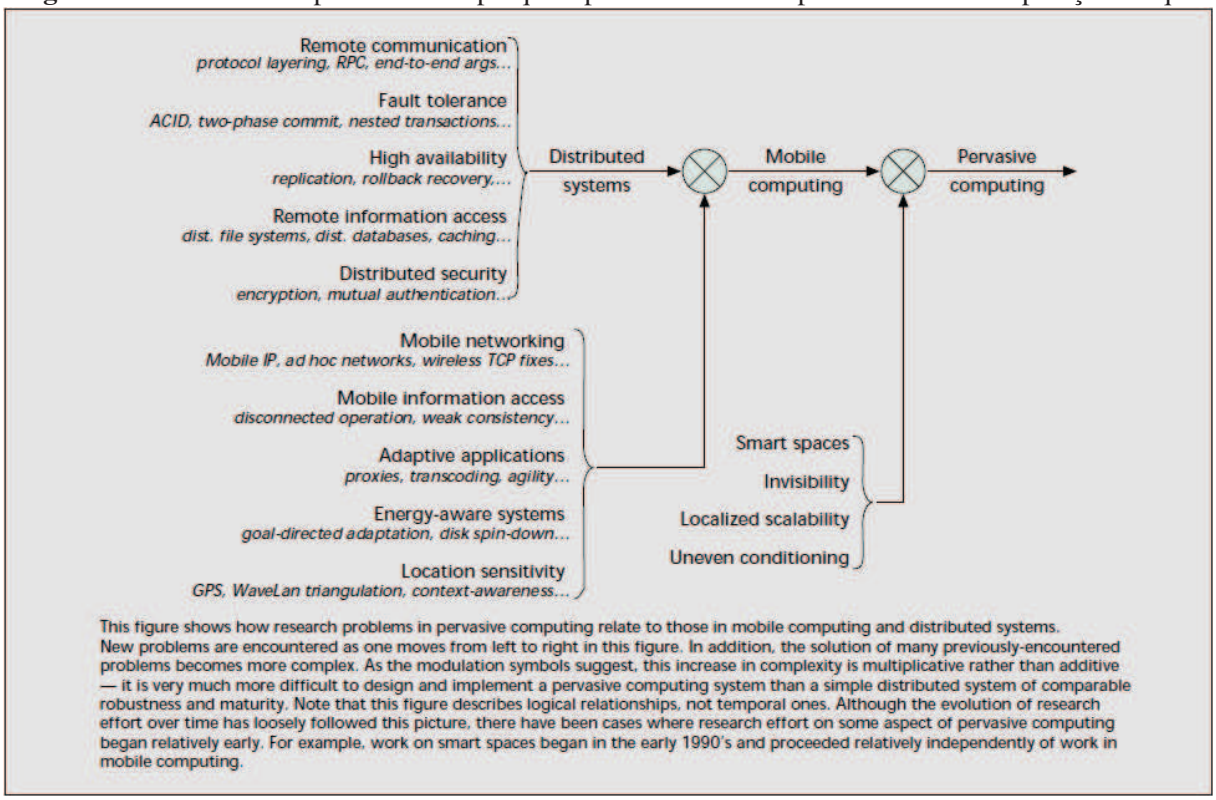
- o tamanho dos computadores estava se tornando cada vez menor;
- o custo do hardware estava se tornando cada vez menor – tendência que continua até os dias atuais;
- a proporção entre o número de pessoas e o número de computadores estava se tornando cada vez maior (em 1960, era comum o uso de um único computador por centenas de pessoas – atualmente, é comum que cada pessoa possua vários dispositivos computacionais, como computadores, GPS, telefones celulares, entre outros);
- o acesso a redes *wireless* estava cada vez mais disponível aos usuários. Embora na época do artigo o acesso *wireless* à Internet ainda não fosse uma realidade para a maioria dos usuários, é evidente que hoje esta também é uma tendência.

Weiser e Brown (1996) concluíram que todas estas tendências ocasionariam um aumento no número de dispositivos e um aumento da disponibilidade de acesso, que eventualmente levaria a uma sobrecarga de informação sobre os usuários, de forma a causar um detrimento na qualidade da interação entre humanos e computadores e, consequentemente, na qualidade de vida das pessoas.

Assim, Weiser (1991) inicia seu artigo afirmando que “as tecnologias mais profundas são aquelas que desaparecem. Elas se entrelaçam no tecido do dia-a-dia até que se tornem indistinguíveis dele”. Ele defende que o futuro da computação está em integrar o computador ao mundo real, e chama este novo paradigma de Computação Ubíqua. A noção de realidade virtual é apresentada como o oposto absoluto da computação ubíqua. Enquanto a realidade virtual busca trazer o usuário para dentro de um mundo construído pelo computador, a computação ubíqua busca trazer os computadores para dentro de um mundo construído por pessoas.

Weiser e Brown (1996) apresentam o conceito de “tecnologia calma” como solução para o problema de sobrecarga de informações sobre o usuário. Este conceito integra-se intimamente com a computação ubíqua. Dentro desta ideia, a transmissão de informações entre o computador e o usuário devem ocorrer na “periferia” da atenção do usuário, movendo-se para o “centro”

Figura 1: Taxonomia de problemas de pesquisa para sistemas computacionais na computação ubíqua



Fonte: (SATYANARAYANAN, 2001)

da atenção em momentos de urgência. Um exemplo é o motor de um carro: quando um motorista dirige um carro, sua atenção está focada na estrada, e ele mal percebe o som do motor. No entanto, se houver algum som diferente no motor (o que pode indicar algum problema), a atenção do usuário passa a focar-se no motor. Da mesma forma, os sistemas devem, em seu curso normal, manter-se na periferia da atenção do usuário, trazendo a atenção para o centro nos momentos em que a atenção é necessária.

Em termos de sistema, Satyanarayanan (2001) vê a computação móvel como evolução da computação distribuída, e a computação ubíqua (ou pervasiva, como ele chama no artigo) como evolução da computação móvel. Ele defende que cada uma destas fases teve seus próprios desafios, e que cada nova fase trouxe junto os desafios da fase anterior, como pode ser visto na Figura 1.

Assim, a computação ubíqua herda os desafios da computação distribuída e móvel, que são: (i) comunicação remota, (ii) tolerância a falhas, (iii) alta disponibilidade, (iv) acesso a informações remotas, (v) segurança distribuída, (vi) redes móveis, (vii) acesso a informações móveis, (viii) aplicações adaptativas, (ix) sistemas sensíveis a energia e (x) sensitividade a localização. A estes problemas de pesquisa, a computação ubíqua acrescenta:

- uso efetivo de espaços inteligentes: à medida que se acrescenta uma infraestrutura computacional aos espaços físicos, torna-se possível não apenas sentir o mundo exterior, mas

também controlá-lo. O contrário também é possível, ou seja, a influência do mundo exterior (como localização) sobre um software do usuário;

- invisibilidade: a busca pelo completo desaparecimento da tecnologia computacional da mente do usuário ou, se inatingível, o mínimo de distração possível;

- escalabilidade localizada: a capacidade de um ambiente suportar uma crescente densidade de dispositivos computacionais, possivelmente através da intensificação das interações locais em detrimento das interações distantes;

- mascaramento de condições irregulares: a capacidade de um sistema de se adaptar a uma taxa de penetração de tecnologia ubíqua variável, dependendo da localização.

2.2 Sensibilidade a Contexto

Satyanarayanan (2001) defende que, para que um sistema seja minimamente intrusivo, é necessário que ele seja capaz de reconhecer a situação do usuário e seus arredores, e adapte seu comportamento de acordo com esta informação. Esta informação é chamada de "contexto" e sistemas capazes deste tipo de adaptação são chamados "sensíveis a contexto".

Uma considerável quantidade de pesquisas tem sido feita na tentativa de obter uma definição exata de contexto. Schilit, Adams e Want (1994) compreendem que uma das principais dificuldades nesta definição está no fato que, na computação móvel (e, por extensão, ubíqua) o ambiente está sempre mudando. A configuração de hardware (dispositivo, qualidade da rede, conectividade), localização, grupo de pessoas próximas, entre outros, está em constante mudança.

Schilit, Adams e Want (1994) definem contexto como sendo: quem você é, com quem você está e que recursos estão próximos. Assim, o contexto é mais do que apenas a localização do usuário; ele inclui também luminosidade, nível de ruído, conectividade de rede, custos de comunicação, qualidade da conexão e até mesmo a situação social, como as pessoas que estão ao redor.

Dey (2001), por sua vez, entende que a definição de Schilit, Adams e Want (1994) é específica demais, pois não encompassa toda a situação relevante à aplicação e seu conjunto de usuários. Como não é possível enumerar todos os aspectos possíveis de todas as situações, a seguinte definição é dada:

> Contexto é qualquer informação que possa ser usada para caracterizar a situação de uma entidade. Uma entidade é uma pessoa, local ou objeto que seja considerada relevante para a interação entre um usuário e uma aplicação, incluindo o próprio usuário e aplicação (DEY, 2001).

Assim, por extensão, Dey (2001) define que "um sistema é sensível a contexto se ele utiliza

o contexto para oferecer informações relevantes e/ou serviços ao usuário, onde a relevância depende da tarefa do usuário".

2.3 Trilhas

Estudos recentes indicam que além do contexto atual de um usuário, o histórico de contextos visitados pode permitir uma ampliação na capacidade de um sistema sensível a contexto de se adaptar a um usuário. A este histórico é dado o nome de Trilha (SILVA et al., 2010; DRIVER; CLARKE, 2008; CLARKE; DRIVER, 2004).

Embora a trilha geralmente esteja vinculada à localização de um usuário, ela possui um conceito mais amplo, que inclui as atividades realizadas, aplicações usadas, conteúdos acessados – entre outros – dentro de um contexto e um período de tempo específico (SILVA et al., 2010).

De acordo com Driver e Clarke (2004), a pesquisa na área de trilhas é relevante para o campo do desenvolvimento de aplicações ubíquas, uma vez que a utilização de trilhas vai ao encontro do desafio central da área de computação ubíqua, isto é, o desenvolvimento de aplicações transparentes ao usuário.

2.4 Ontologias

Sistemas de personalização recentes têm utilizado ontologias como forma de representação de conhecimento. Toda base de conhecimento é baseada em uma conceitualização: os objetos, conceitos e outras entidades que fazem parte de uma área de interesse, e os relacionamentos entre eles (GENESERETH; NILSSON, 1987). Uma conceitualização é uma visão abstrata e simplificada do mundo que se deseja representar para algum propósito. Dentro deste contexto, uma ontologia é uma especificação explícita de uma conceitualização (GRUBER, 1993).

Uma ontologia define um conjunto de primitivas represetacionais que são utilizadas para modelar um domínio de conhecimento. As primitivas representacionais são classes, atributos e relacionamentos. As definições das primitivas representacionais incluem informações sobre seu significado e restrições em sua aplicação logicamente consistente (LIU; ÖZSU, 2009).

As ontologias são consideradas o método de armazenamento de conhecimento mais rico para a modelagem de contextos, levando em consideração os seguintes requerimentos: (i) composição distribuída, (ii) validação parcial, (iii) riqueza e qualidade de informação, (iv) resolução de ambiguidade, (v) nível de formalidade e (vi) aplicabilidade a ambientes existentes (STRANG; LINNHOFF-POPIEN, 2004). A formalização de entidades reais em dados compreensíveis por máquina facilita o compartilhamento de conhecimento contextual e reuso de informação. Desta forma, computadores podem determinar a compatibilidade contextual, comprar fatos contextuais e inferir informações de contexto novas e mais completas (KRUMMENACHER; STRANG, 2007).

2.5 Personalização

Mulvenna, Anand e Büchner (2000) definem Personalização como sendo a disponibilização de produtos, serviços e informações adaptados individualmente. A tecnologia de personalização geralmente envolve programas que aprendam padrões, hábitos e preferências de um usuário.

Silva et al. (2010) definem Entidade como uma extensão do conceito de usuário. Enquanto um usuário geralmente refere-se a uma pessoa que está utilizando um aplicativo, a definição de entidade utilizada por estes autores é de que pode ser qualquer entidade, como uma pessoa, um carro ou mesmo um dispositivo.

Dentro do âmbito da personalização, este trabalho dá à estrutura de dados utilizada para armazenar as informações das entidades por uma aplicação o nome de Modelo de Entidade, enquanto o conjunto de informações referentes a uma entidade específica é nomeada Perfil de Entidade.

A coleta de informações de uma entidade pode ser explícita ou implícita (VIVIANI; BENNANI; EGYED-ZSIGMOND, 2010). Na coleta explícita, é requisitado ao usuário que preencha informações através de um formulário pré-definido, enquanto na coleta implícita, a informação é inferida a partir do estudo do comportamento da entidade.

O objetivo principal por detrás dos sistemas de personalização modernos é oferecer às entidades o que elas querem sem requisitar que elas informem isto de forma explícita (MULVENNA; ANAND; BüCHNER, 2000). Fica claro que este propósito vai ao encontro do propósito da computação ubíqua, isto é, o desenvolvimento de uma tecnologia que "desapareça".

Informações a respeito de uma entidade colhidas por uma aplicação podem ser úteis para outras aplicações utilizadas pela mesma entidade (VIVIANI; BENNANI; EGYED-ZSIGMOND, 2010). No entanto, os diferentes modelos de entidade utilizados pelas aplicações tornam difícil a utilização de informações de um mesmo perfil de entidade por várias aplicações. Além disso, algumas das informações da entidade podem ser válidas apenas dentro de um contexto específico, o que dificulta ainda mais que diferentes aplicações compartilhem o mesmo perfil.

Visando resolver estas questões, o conceito de personalização inter-sistemas foi introduzido (NIEDERéE et al., 2004) como forma de compartilhar informações de entidades entre diferentes aplicações. Como o maior problema deste tipo de modelo é o alinhamento entre as diferentes abordagens de personalização, o estudo de sistemas de modelos de usuários (e, por extensão, de entidades, dentro do âmbito deste trabalho) é atualmente o principal desafio da pesquisa na área de personalização (VIVIANI; BENNANI; EGYED-ZSIGMOND, 2010).

O *IEEE Standard Computer Dictionary* (GERACI, 1991) define interoperabilidade como "a capacidade de dois ou mais sistemas ou componentes trocarem informações e utilizarem as informações que foram trocadas". No âmbito da personalização, interoperabilidade é a capacidade de acessar e interpretar informação derivada de múltiplas fontes heterogêneas e integrar esta informação em um modelo de entidade comum. Os dois principais modelos de interoperabilidade são (CARMAGNOLA; CENA, 2009):

- interoperabilidade sintática: onde as diferenças entre os modelos de entidade são resolvidos em nível de aplicação;
- interoperabilidade semântica: onde as diferenças entre os modelos são resolvidos em nível de conhecimento.

As duas principais abordagens utilizadas em sistemas de modelos de entidades interoperáveis são (VIVIANI; BENNANI; EGYED-ZSIGMOND, 2010):

- baseados em padrões: são baseados numa visão *top-down*, definindo um padrão que todas as aplicações envolvidas devem observar;
- baseados em mediação: se baseiam numa visão *bottom-up*, onde ocorre algum tipo de reconciliação entre os diferentes modelos de representação.

Diferentes trabalhos definem perfis de diferentes formas, como um registro de interesses negativos e positivos em itens específicos (WASFI, 1999), uma representação estruturada das necessidades do usuário (AMATO; STRACCIA, 1999), ou qualquer informação a respeito de um indivíduo que seja essencial para a aplicação que está sendo considerada (SCHIAFFINO; AMANDI, 2009). No contexto deste trabalho, um perfil foi definido como um conjunto de informações relevantes a respeito de uma entidade, processado a partir das decisões tomadas por esta entidade.

2.6 Considerações sobre o Capítulo

Os conceitos de computação ubíqua, sensibilidade a contexto, trilhas e personalização se complementam na busca de conhecer quem a entidade é. Através da computação ubíqua, é possível acompanhar os passos e ações da entidade. Através da sensibilidade a contexto, é possível compreender o ambiente da entidade e a influência que este ambiente tem sobre ela. Através das trilhas, é possível analisar o histórico de uma entidade dentro de contextos visitados anteriormente. E a personalização torna possível disponibilizar estas informações de forma simples através de um perfil único, que pode ser consultado e utilizado na tomada de decisões de um sistema.

O próximo capítulo discute trabalhos relacionados ao gerenciamento de perfis.

3 TRABALHOS RELACIONADOS

Esta seção apresenta trabalhos relacionados com o modelo proposto, bem como uma comparação entre os mesmos. Foram escolhidos trabalhos que tivessem como foco a adaptação de um perfil de entidade de acordo com as ações tomadas pela mesma. Como não foi encontrado nenhum modelo que tivesse o foco pretendido para este trabalho (montagem de um perfil através de inferência em trilhas), foram escolhidos modelos que tivessem uma ou mais características que se farão úteis para o modelo deste trabalho.

Os trabalhos apresentados utilizam o conceito de modelo e perfil de usuário ao invés de modelo e perfil de entidade, que é o termo utilizado neste trabalho. Assim sendo, os termos modelo e perfil de usuário serão utilizados quando apropriado.

3.1 Kim e Lee (2008)

Kim e Lee (2008) apresentam um *framework* para gerenciamento de perfis baseado em *Web Services* para fornecer serviços personalizados. Os autores compreendem gerenciamento de perfil como sendo a coleta de um conjunto de informações necessárias para prover aos usuários cinco tipos de serviço personalizado: (i) configuração de perfil, (ii) criação de perfil, (iii) troca de perfil, (iv) processamento de perfil e (v) serviço de perfil.

O *framework* é montado de acordo com a Figura 2. A comunicação entre os módulos é feita através de *Web Services*, de modo a poder operar de forma distribuída entre várias plataformas e dispositivos.

A configuração de perfil configura os perfis em dois tipos: perfis básicos, requeridos para todos os tipos de serviço, e perfis de serviço, configurados diferentemente para cada tipo de serviço. O tipo de perfil define o tipo de informação que irá conter cada perfil.

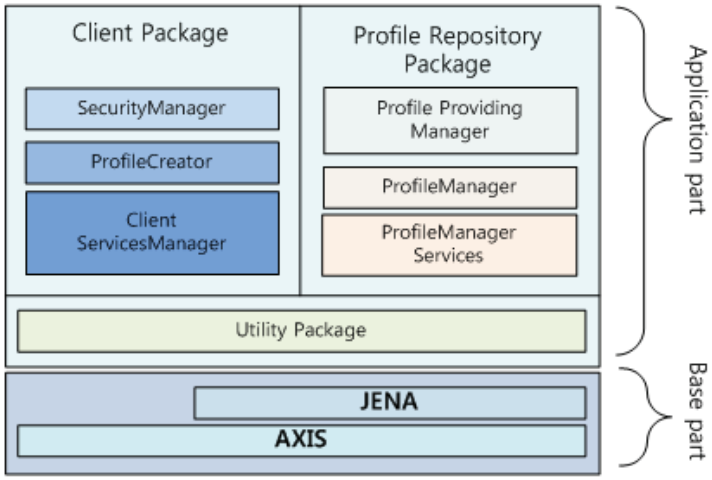
A criação de perfil é a tecnologia que permite que clientes e provedores de serviço criem os perfis. A criação dos perfis é feita através de um conjunto de componentes, de acordo com a Figura 3. O gerenciador de criação cria o perfil, com base nos arquivos de *template* de perfil, que é então codificado pelo gerenciador de segurança, num processo coordenado pelo gerenciador de serviços de perfil. Toda a comunicação entre os módulos ocorre via *Web Services*.

Troca de perfis é o processo através do qual um perfil é enviado pelo servidor ao cliente. O processamento do perfil classifica os perfis transferidos pelos clientes através de regras e os armazena em uma base de dados. Os serviços de perfil extraem os perfis do repositório e os fornecem aos provedores de serviço quando os mesmos os requisitam.

3.2 Chellouche, Arnaud e Négru (2010)

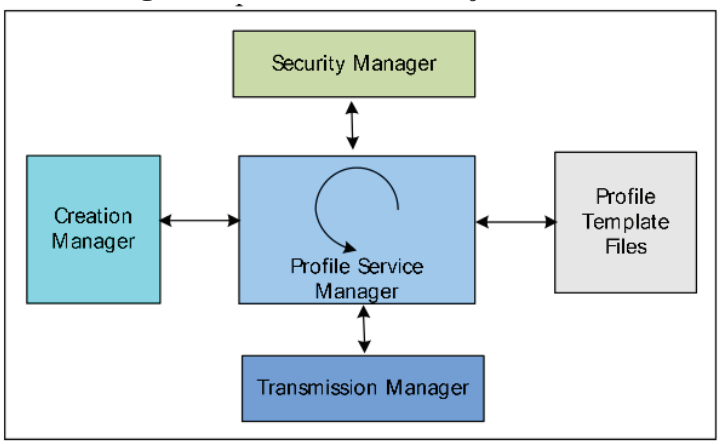
O modelo de Chellouche, Arnaud e Négru (2010) busca resolver as dificuldades atuais no desenvolvimento de aplicativos para dispositivos móveis, uma vez que os mesmos possuem

Figura 2: Framework de perfis



Fonte: (KIM; LEE, 2008)

Figura 3: Mecanismo de Criação de Perfis



Fonte: (KIM; LEE, 2008)

uma grande variedade de resolução, memória, poder computacional, entre outros. Além disso, a qualidade da conexão de um usuário pode variar significativamente, e isto também deve ser levado em consideração no desenvolvimento.

O conjunto de informações que caracteriza um usuário é chamada de perfil de usuário. Com a intenção de compartilhar o contexto do usuário entre serviços e minimizar as mudanças no desenvolvimento de aplicações sensíveis a contexto, os autores propõem como solução um *middleware* que gerencia o perfil de um usuário dinamicamente e supre às aplicações todas as informações que elas podem precisar durante o período de adaptação.

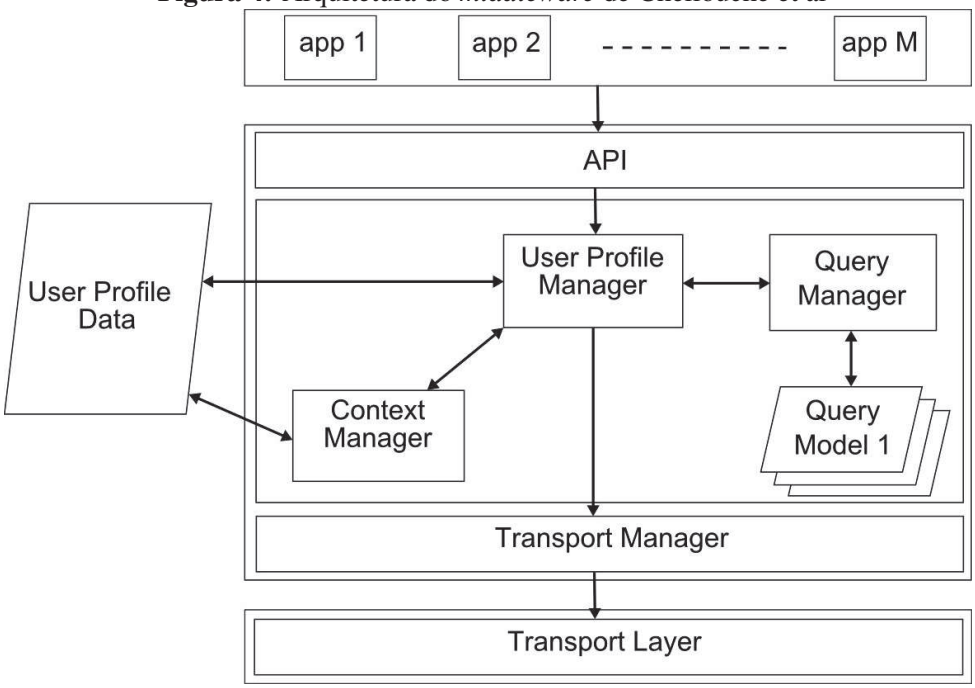
O perfil do usuário é constituído por seu contexto, onde a definição de contexto utilizada é aquela dada por Dey (2001), na qual contexto refere-se a qualquer informação que pode ser utilizada para caracterizar a situação de uma entidade. Assim, o perfil de usuário é dividido em cinco componentes:

- Perfil geral: refere-se às informações básicas, como nome, endereço, telefone, idade, sexo, entre outros;
- Perfil do dispositivo: contém informações a respeito dos dispositivos do usuário, tais como marca, capacidade, sistema operacional, entre outros;
- Perfil de rede: descreve as redes às quais o usuário pode se conectar, com suas características;
- Perfil dos serviços: registra as informações a respeito de serviços, como nome, versão, porta e conteúdo;
- Perfil de contexto: contém informações a respeito do ambiente do usuário, como hora, data e localização, aplicações em execução e banda disponível. É considerada a parte dinâmica do perfil, composta por dados voláteis.

O *middleware* é apresentado segundo ilustrado na Figura 4, sendo composto dos seguintes módulos:

- Gerenciador de perfil de usuário: módulo central da arquitetura, que interage com todos os outros módulos com o objetivo de construir o perfil correto;
- Gerenciador de contexto: monitora o ambiente, coletando informações com o objetivo de atualizar a parte dinâmica do perfil;
- Gerenciador de consultas: módulo responsável por validar e salvar um modelo de consulta para cada aplicação;
- Gerenciador de trasporte: faz a comunicação do perfil de usuário para a aplicação;
- API: constitui o ponto de interface entre as aplicações e o *middleware.*

Figura 4: Arquitetura do *middleware* de Chellouche et al



Fonte: (CHELLOUCHE; ARNAUD; NéGRU, 2010)

O perfil é construído seguindo o fluxo apresentado na Figura 5. A aplicação solicita os dados ao gerenciador de perfis, através da API, que por sua vez repassa a solicitação ao gerenciador de consultas. Ao mesmo tempo, o gerenciador de contextos é informado que a aplicação está em execução e passa a atualizar o gerenciador de perfis com as informações atualizadas. O perfil de usuário é criado e enviado ao gerenciador de transporte, que retorna o perfil solicitado à aplicação solicitante.

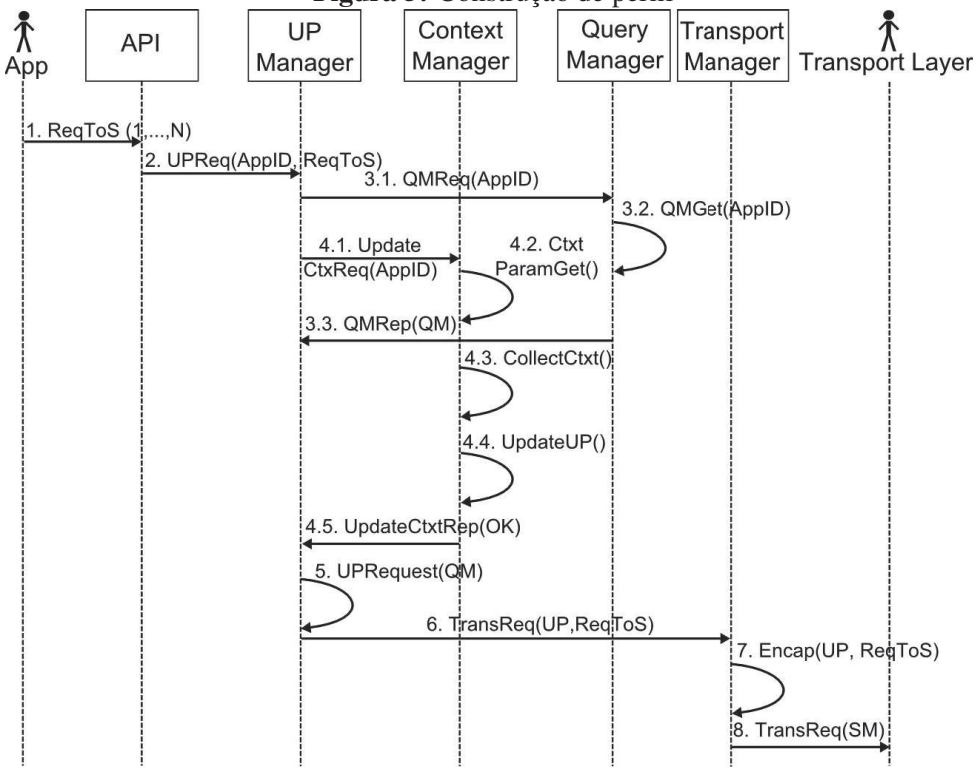
3.3 Niederée et al. (2004)

O artigo apresentado por Niederée et al. (2004) busca responder três indagações, que são apresentadas como sendo os três desafios básicos da personalização:

- como modelar o usuário e sua situação de modo extensível e unificado, de forma que este modelo possa ser interpretado e usado para personalização por diferentes sistemas?
- como coletar dados para preencher e atualizar perfis de usuário a partir das interações do usuário com diferentes sistemas baseados em um modelo de contexto de usuário unificado?
- como habilitar conceitualmente e tecnicamente diferentes sistemas a fazer uso dos perfis de usuário coletados para aperfeiçoamento do suporte ao usuário?

Os autores apresentam como resposta a estas indagações um modelo denominado UUCM, que tem como objetivo modelar características do usuário e de seu contexto. O conceito de

Figura 5: Construção de perfil



Fonte: (CHELLOUCHE; ARNAUD; NéGRU, 2010)

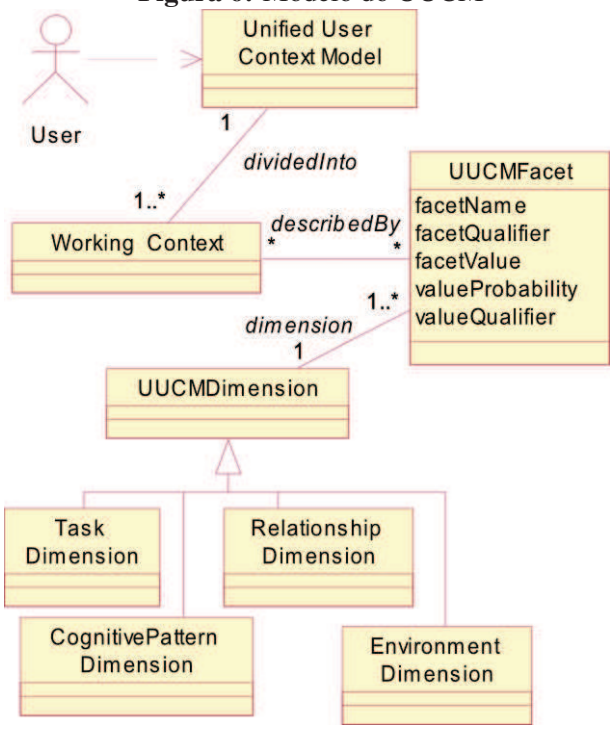
contexto utilizado é aquele apresentado por Benerecetti, Bouquet e Ghidini (2000), no qual o contexto pode ser pensado como as informações extras, geralmente implícitas (como associações, fatos, suposições), que podem ser utilizadas para compreender plenamente uma interação, comunicação ou representação de conhecimento.

A abordagem para a modelagem unificada de contextos de usuário é proposta em dois níveis. No nível **abstrato**, são definidos os blocos básicos de construção: contexto de usuário, facetas do modelo de usuário, propriedades principais e dimensões do modelo de usuário. O termo faceta representa as diferentes características do usuário. Este nível define um metamodelo para as dimensões e facetas concretas utilizadas na descrição do modelo de contexto de usuário. Para que a abordagem de personalização multi-sistema seja válida, é necessário que este metamodelo de contexto de usuário seja publicado como uma ontologia compartilhada, e que todos os participantes utilizem este modelo.

No nível **concreto**, um conjunto de dimensões e facetas são definidos. As facetas e dimensões são descritas como parte de uma ontologia adicional que é compartilhada pelos componentes envolvidos na personalização multi-sistemas.

O modelo do UUCM é apresentado na Figura 6. Segundo o modelo, um perfil de usuário (*Unified User Context Model*) é dividido em um conjunto de contextos de trabalho (*Working Context*), onde cada contexto é descrito por um conjunto de facetas (*UUCM Facet*). Cada faceta é vinculada a uma dimensão (*UUCMDimension*).

Figura 6: Modelo do UUCM



Fonte: (NIEDERéE et al., 2004)

O UUCM apenas define o método através do qual o perfil de contexto de usuário é descrito e estruturado para personalização multi-sistema. Para a descrição de perfis de contexto concretos, o UUCM utiliza ontologias (ou vocabulários) para as dimensões, facetas e valores de faceta.

As seguintes dimensões são utilizadas:

- Cognitiva (*CognitivePattern*): descreve as dimensões cognitivas do usuário, tais como áreas de interesse, competências e preferências;
- Tarefa (*Task*): descreve os objetivos do usuário quando ele interage com sistemas. Esta dimensão utiliza dados tais como tarefa atual, papel do usuário na tarefa e histórico da tarefa;
- Relacionamentos (*Relationship*): armazena as entidades com as quais o usuário tem alguma relação;
- Ambiente (*Environment*): parâmetros que são tipicamente utilizados para abordagens sensíveis a contexto, tais como tempo, localização, dispositivo e linguagem.

O perfil de usuário do UUCM é codificado como um RDF Schema(W3C, 2011a), ampliado com expressões OWL.

O modelo inclui ainda o conceito de Passaporte de Contexto. O passaporte de contexto é uma representação compacta do modelo de contexto atual do usuário para a personalização multi-sistemas. Ele contém informações ontologicamente arranjadas a respeito dos padrões

cognitivos (habilidades, interesses, entre outros), o ambiente (data, local, dispositivo usado) e sua rede pessoal de pessoas e relacionamentos envolvidos.

Para utilizar o passaporte de contexto em um ambiente de personalização multi-sistemas, o usuário "apresenta" o passaporte ao sistema que o usuário deseja utilizar. Como o passaporte está vinculado a uma ontologia compartilhada, é possível que o sistema consiga interpretar parcialmente o passaporte através de uma arquitetura mediadora. Como resultado, dois fluxos são possíveis: (i) do passaporte para o sistema, auxiliando o sistema a compreender o contexto do usuário, e (ii) do sistema para o passaporte, realizando alterações nas informações de contexto do usuário.

3.4 Levis et al. (2008)

O projeto PeLeP, apresentado por Levis et al. (2008), integra a utilização de perfis aperfeiçoados automaticamente com ambientes de educação ubíqua.

Barbosa, Geyer e Barbosa (2005) compreendem Educação Ubíqua como sendo processos educacionais que ocorrem em qualquer lugar, a qualquer tempo e com qualquer dispositivo, de forma contínua, contextualizada e integrada ao cotidiano do aprendiz. Desta forma, o conhecimento presente no dia-a-dia das mais diferentes formas e em diferentes locais pode ser relacionado com processos educacionais direcionados aos aprendizes.

O PeLeP, cujo modelo é mostrado na Figura 7, é dedicado ao gerenciamento de perfis de aprendizes em um ambiente ubíquo de ensino e aprendizagem. O modelo administra os perfis automaticamente, inferindo informações através do histórico do aprendiz. As alterações realizadas no perfil refletem o comportamento do aprendiz no ambiente ubíquo, levando em consideração o seu modelo de mobilidade e de contexto.

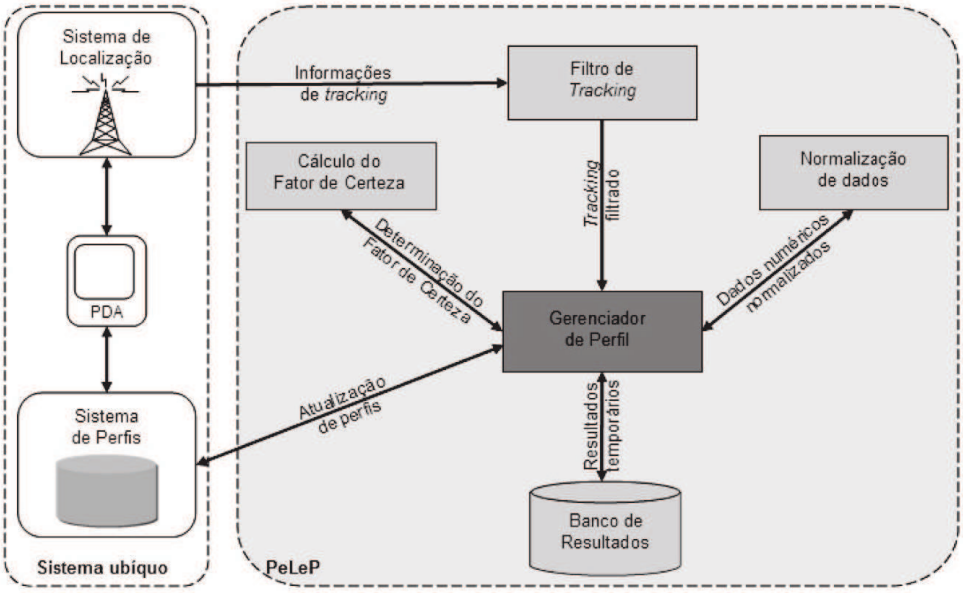
No PeLeP, o Perfil do Aprendiz consiste em um perfil contendo campos pré-definidos, voltados para o ambiente de educação ubíqua. Os campos são distribuídos nas seguintes categorias: identificação, objetivos, agenda, segurança, relacionamentos, competências e preferências.

É realizado um acompanhamento (*tracking*) do histórico do aprendiz, onde são armazenados os contextos visitados pelo usuário, bem como os conteúdos, dispositivos e aplicativos utilizados em cada contexto. Os valores relativos das variáveis numéricas recebidas no *tracking* são regularizados através de um processo de normalização de amplitude. Como diferentes valores podem ter diferentes impactos para a determinação da preferência de um usuário, calcula-se o Fator de Certeza (FC), de acordo com a equação 3.1, onde f_i é o valor normalizado para o fator, p_i o seu percentual, e n o número de fatores considerados.

$$FC = \frac{\sum_{i=0}^{n} f_i p_i}{\frac{100}{n}} \tag{3.1}$$

O fator tempo também é levado em consideração, onde Fatores de Certeza mais antigos têm

Figura 7: Arquitetura do PeLeP



Fonte: (LEVIS et al., 2008)

um impacto menor. Assim, por exemplo, se o usuário trocar o seu aplicativo preferido, à medida que o tempo for passando o impacto do aplicativo anterior será cada vez menor.

O processo de atualização do modelo do aprendiz é então realizado através de regras pré-definidas, que levam em conta as ocorrências e o seu fator de certeza. As regras suportam conclusões a partir de evidências antecedentes, e cada categoria de informação possui regras diferentes.

O PeLeP foi validado através da integração com o LOCAL (*Location and Context-Aware Learning*), um sistema de educação ubíqua sensível a contexto (BARBOSA et al., 2011).

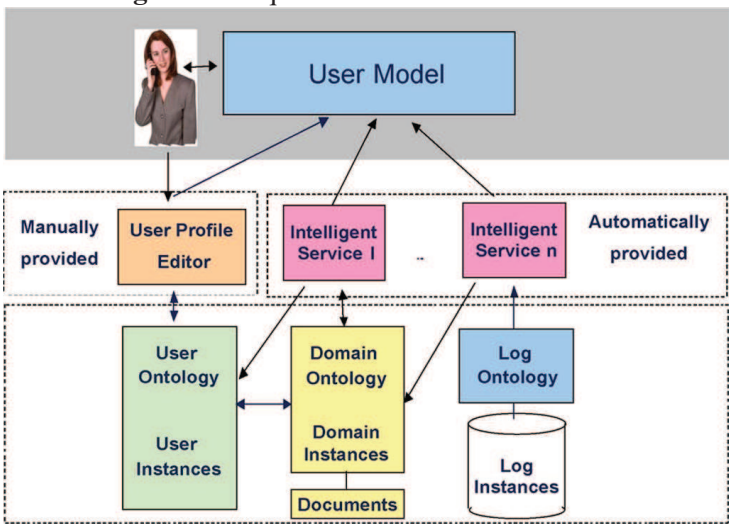
3.5 Razmerita (2011)

Razmerita (2011) apresenta o modelo OntobUMf (*Ontology-based User Modeling framework*), um *framework* para modelagem de usuários baseado em ontologia. O artigo defende que o gerenciamento de conhecimento (KM, ou *Knowledge Management*) é uma tarefa importante para indivíduos e organizações, e que sistemas de gerenciamento de conhecimento (KMS) atuais focam no compartilhamento e na criação de conteúdo que, por sua vez, leva à inovação tanto a nível individual quanto corporativo. Mais recentemente, o KM pessoal tem se tornado relevante, e foca na busca do indivíduo em encontrar, aprender, compartilhar conteúdo e socializar.

O modelo OntobUMf busca unir o KM tradicional com o KM pessoal através da personalização de um KMS, de modo a permitir a customização de *interfaces* e adaptação da funcionalidade, estrutura, conteúdo e modularidade, aumentando assim a relevância para usuários individuais.

A Figura 8 apresenta a arquitetura do modelo OntobUMf. OntobUMf é uma aplicação de

Figura 8: Arquitetura do modelo OntobUMf



Fonte: (RAZMERITA, 2011)

três camadas: (i) *front end*, (ii) serviços e (iii) dados. A camada de *front end* é um conjunto de *interfaces* que permitem que o usuário acesse e atualize seu modelo de usuário ou perfil de usuário.

A camada de serviços contém serviços inteligentes, tais como algoritmos de modelagem de usuários, mecanismos de personalização, entre outros. Os serviços possuem duas tarefas principais no sistema. A primeira é atualizar e manter o modelo de usuário tendo por base os dados de uso disponíveis do sistema através de um extrator de categoria, que é composto por um conjunto de mecanismos para modelagem das características dos usuários que estão interagindo com o sistema. A segunda tarefa é fornecer um conjunto de serviços personalizados baseados nas características dos usuários.

A camada de dados inclui dados de usuários relacionados a dados específicos de domínio. Ontologias capturam o conceito e as relações entre os conceitos descrevendo diferentes recursos disponíveis no sistema. OntobUMf possui três ontologias. A ontologia de usuários estrutura as características dos usuários, classificando-as em conceitos, subconceitos, propriedades e relacionamentos. A ontologia de domínios descreve o domínio de conhecimentos e os relacionamentos entre os diferentes conceitos. A ontologia de log define a semântica de interação do usuário com o sistema, descrevendo as ações do usuário e os eventos associados executados pelo sistema.

A OntobUMf classifica os usuários de acordo com o nível de atividade, o tipo de atividade e nível de compartilhamento de conhecimento. Baseado no tipo de atividade, os usuários são classificados como “leitores”, “escritores” ou “espiões”. Baseado no nível de atividade, eles são classificados como “muito ativo”, “ativo”, “visitante” e “inativo”. Baseado no nível de compartilhamento de conteúdo, eles são classificados como “inconsciente”, “consciente”, “interessado”, “tentativo” ou “adotante”.

3.6 Tao, Li e Zhong (2011)

No artigo apresentado por Tao, Li e Zhong (2011), os autores avaliam que, embora ontologias venham sendo cada vez mais utilizadas como modelo de descrição e formalização de conhecimento, a maioria dos modelos utiliza somente as informações de uma base de conhecimento global ou informações locais de usuário para montar um perfil de usuário. Desta forma, é proposto um modelo de ontologia sobre perfis de usuários que compõe-se tanto a partir de uma base de conhecimento global quanto de repositórios locais de usuários.

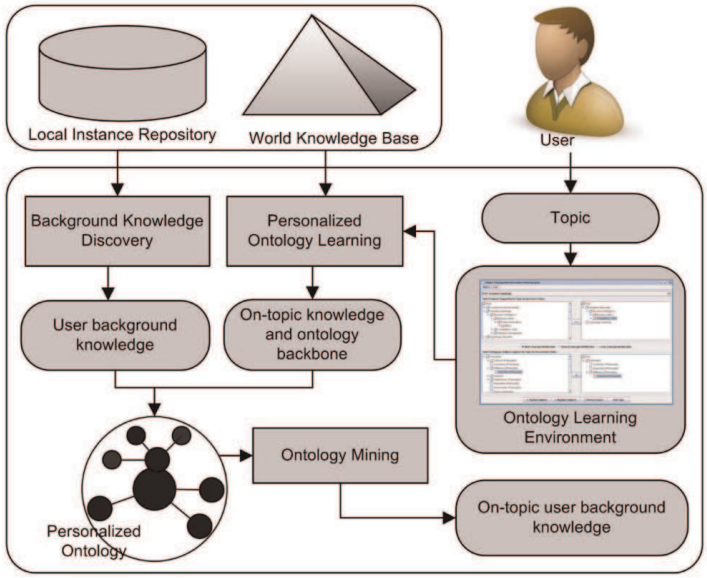
De acordo com os autores, ontologias personalizadas são um modelo de conceitualização que formalmente descrevem e especificam o conhecimento de fundo do usuário. Por exemplo, quando o usuário busca pelo tópico "New York" em um agente de buscas, a informação demandada é diferente para usuários de negócios e turistas. A expectativa pode ser diferente até mesmo para o mesmo usuário fazendo a mesma busca em contextos diferentes (um usuário de negócios pode estar fazendo a busca para uma viagem de negócios ou a lazer, por exemplo). Desta forma, o modelo conceitual do usuário pode mudar de acordo com a sua necessidade.

A primeira base de informação é uma representação de conhecimento global. A base utilizada pelo modelo é a *Library of Congress Subject Headings* (LCSH). A LCSH foi desenvolvida para organizar e consultar informações de um grande volume de coleções da Biblioteca do Congresso Americano. O sistema da LCSH possui três tipos de referência: "termo amplo", "usado para" e "termo relacionado". Estas referências e as ligações entre elas são usadas para construir o conhecimento de mundo necessário para a composição da ontologia.

A base de interesses de usuários é extraída através de uma ferramenta denominada *Ontology Learning Environment* (OLE). A ferramenta constrói a ontologia através da alimentação manual de informações pelo usuário. A ontologia classifica os assuntos alimentados pelo usuário em três tipos: (i) "assuntos positivos", (ii) "assuntos negativos" e "assuntos neutros." Assuntos positivos são conceitos relevantes para a informação necessária, e assuntos negativos são conceitos que resolvem interpretações ambíguas ou paradoxais. Assuntos neutros são aqueles que não tem relevância para a informação buscada.

A Figura 9 apresenta a arquitetura do modelo. O passo inicial do processo é a construção da ontologia, de acordo com um determinado tópico, utilizando como base as duas fontes de conhecimento (base de conhecimento global e as informações locais de usuário). A base de conhecimento local e as informações providas pelo OLE são utilizadas para gerar o conjunto de informações sobre o assunto que, unido ao *background* do usuário, são processados através de um processo chamado "mineração de ontologia", que gera o conhecimento de *background* do usuário a respeito de um determinado tópico.

Figura 9: Arquitetura do modelo de Tao, Li e Zhong



Fonte: (TAO; LI; ZHONG, 2011)

3.7 Considerações sobre os trabalhos relacionados

A Tabela 1 exibe um comparativo entre os trabalhos apresentados neste Capítulo. Os seguintes fatores são levados em consideração:

- Interoperabilidade: considera se o modelo apresenta interoperabilidade – caso positivo, se a mesma é sintática ou semântica – de acordo com a definição apresentada na Seção 2.5;
- Modelos dinâmicos: identifica se o modelo permite, de alguma forma, que sejam alterados os modelos de entidade (ou usuário) ou as regras para geração de perfis durante a execução;
- Armazenamento: define onde ficam armazenados os perfis do usuário;
- Sensibilidade a contexto: considera se o contexto do usuário é levado em consideração na formação do perfil;
- Utiliza trilhas: define se as trilhas do usuário são levadas em consideração na construção do perfil. Na tabela, a definição de "Parcial" significa que o modelo usa algum tipo de histórico para a montagem do perfil, mas não utiliza a definição formal de trilhas;
- Domínio: identifica se o perfil é voltado para um domínio específico, ou se o mesmo permite que vários domínios possam ser utilizados.

Tabela 1: Comparação entre os trabalhos apresentados

Trabalho	Interope-rabilidade	Modelos dinâmicos	Armaze-namento	Sensibilidade a contexto	Utiliza trilhas	Domínio
Kim e Lee (2008)	Sintática	Não	Servidor	Não	Não	Genérico
Chellouche, Arnaud e Négru (2010)	Sintática	Não	Servidor	Sim	Não	Multimídia
Niederée et al. (2004)	Sintática	Não	Cliente	Sim	Não	Genérico
Levis et al. (2008)	Não	Não	Servidor	Sim	Parcial	Educação
Razmerita (2011)	Sintática	Não	Servidor	Sim	Parcial	Genérico
Tao, Li e Zhong (2011)	Semântica	Não	Servidor	Sim	Parcial	Genérico

Fonte: elaborado pelo autor

Os trabalhos apresentam uma série de características úteis para o modelo exposto neste trabalho. A maior parte dos modelos apresentados realiza algum tipo de aperfeiçoamento automático de perfis baseado nas ações do usuário. Enquanto alguns trabalhos atualizam os perfis no momento da realização da ação do usuário ou entidade, outros armazenam estes eventos em um histórico, e realizam a atualização através da análise do histórico.

A maior parte dos modelos armazena os perfis em um servidor, enquanto um deles armazena no cliente. O armazenamento em um servidor traz vantagens e desvantagens. A principal desvantagem é a indisponibilidade do perfil no caso de falta de conectividade entre o servidor e o cliente. No entanto, o armazenamento em um servidor aumenta a segurança da informação, permite um maior poder de processamento e simplifica o modelo.

Enquanto alguns trabalhos se especializam em domínios específicos (tais como multimídia ou educação), outros modelos permitem seu uso em qualquer domínio. A especialização simplifica o modelo, mas o domínio genérico expande a possibilidade de utilização e, especialmente, de interoperabilidade.

Enquanto alguns modelos apresentaram a utilização do histórico do usuário para a montagem do perfil, nenhum deles utilizou o conceito de trilha, o que enriqueceria ainda mais a geração do perfil pois permitiria a utilização de eventos contextualizados. A trilha, ao contrário do histórico, possui não apenas informações de eventos realizados, mas também o contexto que cerca estas ações. Além disso, em todos os trabalhos, as regras de processamento de perfil e o modelo de perfil foram apresentados como estáticos, isto é, configuráveis no início da execução mas não posteriormente. Como estas são características relevantes e permitem a geração de informações de perfil mais precisas e completas, este trabalho busca preencher esta lacuna.

3.8 Considerações finais

Este capítulo apresentou um conjunto de propostas focadas na personalização através da adaptação de perfis de entidade de acordo com as ações tomadas pelas mesmas. A Seção 6.2 apresenta uma comparação entre os trabalhos apresentados neste capítulo e o modelo que está sendo proposto nesta dissertação.

4 MODELO PROPOSTO

Este capítulo apresenta o modelo **eProfile**. A Seção 4.1 descreve o funcionamento do modelo. A Seção 4.2 detalha a arquitetura do eProfile. A Seção 4.3 apresenta os agentes implementados. A Seção 4.4 apresenta as ontologias usadas por este modelo, e a Seção 4.5 demonstra como são implementadas as regras utilizadas para inferir estas ontologias. A Seção 4.6, apresenta um exemplo de operação, com a Seção 4.7 apresentando as considerações sobre o Capítulo.

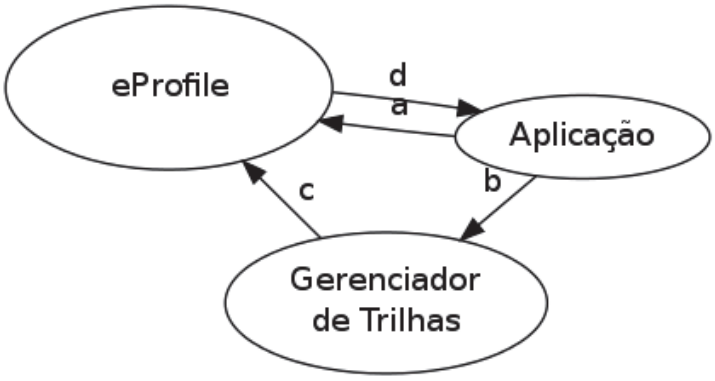
4.1 Modelo eProfile

Segundo Weiser e Brown (1996), é desejável que os sistemas do futuro se tornem cada vez mais invisíveis aos seus usuários, e Satyanarayanan (2001) defende que um sistema só pode se tornar invisível se for proativo, e só pode ser proativo se conhecer o usuário. Para que um sistema possa conhecer uma entidade, ele deve conhecer o histórico da entidade, compreender o contexto em que ela está inserida, bem como os dados coletados a respeito dela no mesmo contexto (ou num contexto similar), e ser capaz de processar este conjunto de informações em um perfil de entidade.

A Figura 10 apresenta a visão geral do eProfile, onde cada círculo representa um componente, e as setas indicam o fluxo dos dados.

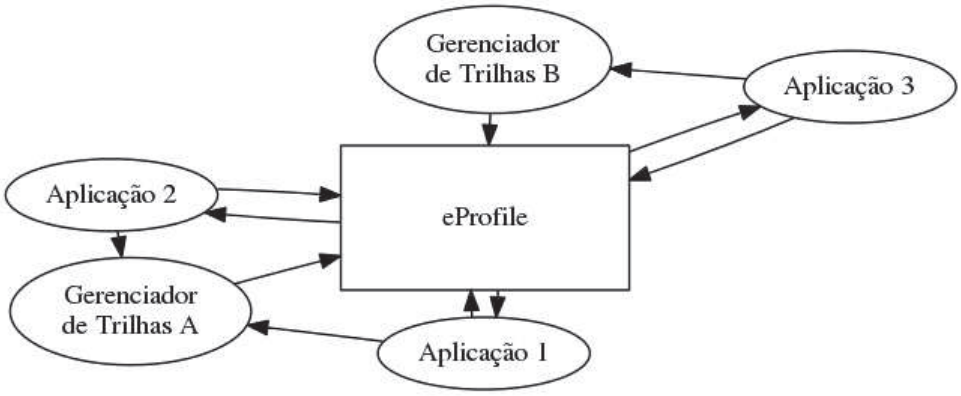
O primeiro componente representado é a **aplicação**. A aplicação, dentro deste modelo, é qualquer aplicação que está sendo utilizada diretamente por uma entidade. Para que uma aplicação seja considerada apta a este modelo (isto é, possa utilizar as funcionalidades oferecidas pelo eProfile), ela precisa ser capaz de registrar suas informações em um gerenciador de trilhas suportado pelo eProfile, e deverá seguir o protocolo de comunicação do eProfile para se comunicar com o mesmo. Seguidos estes pré-requisitos, a aplicação pode ser de qualquer tipo e operar em qualquer tipo de dispositivo que tenha comunicação com a internet.

Figura 10: Visão geral do eProfile



Fonte: elaborado pelo autor

Figura 11: eProfile conectado a mais de um gerenciador de trilhas



Fonte: elaborado pelo autor

O segundo componente representado é um **gerenciador de trilhas**. Um gerenciador de trilhas é capaz de registrar eventos realizados por uma entidade, colocando-os em uma ordem cronológica e registrando informações relevantes ao evento (como o contexto na qual o evento foi realizado). O modelo do eProfile é desenhado para oferecer suporte a qualquer gerenciador de trilhas, desde que o protocolo de comunicação entre o gerenciador de trilhas e o eProfile esteja codificado no eProfile.

A Figura 10 apresenta ainda um terceiro componente, o eProfile, que é o modelo apresentado neste trabalho. Detalhes sobre a arquitetura do eProfile são apresentados na Seção 4.2.

As setas indicam o fluxo dos dados. A comunicação tem início quando a aplicação registra-se no eProfile (seta “a”), enviando (i) o modelo de entidade utilizado pela aplicação e (ii) as regras para construção deste modelo. A partir deste momento, a aplicação passa a registrar os eventos no gerenciador de trilhas (seta “b”). O eProfile periodicamente busca esta informação a partir do gerenciador de trilhas, e constrói ou reconfigura o perfil de entidade (seta “c”). À medida que a aplicação tem necessidade de um perfil atualizado, ela busca este perfil no eProfile (seta “d”).

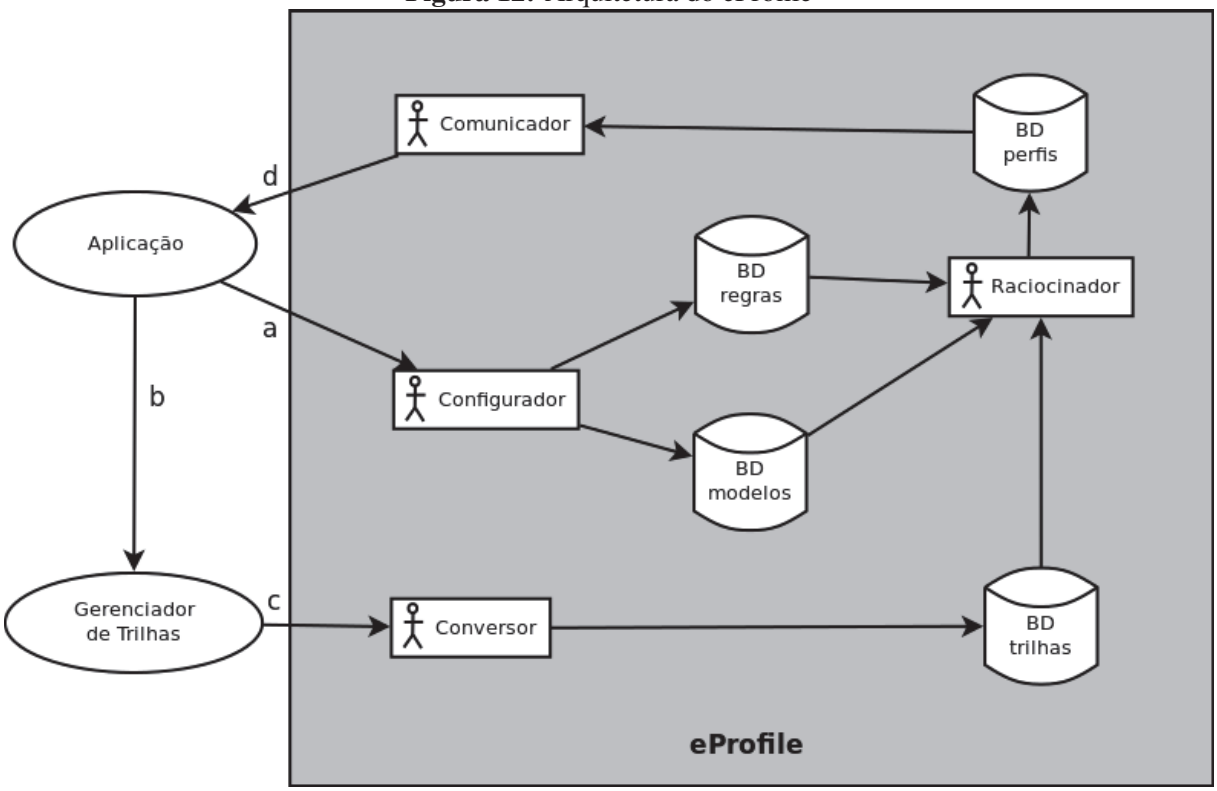
Como o eProfile oferece suporte a várias aplicações, é possível que diferentes aplicações utilizem diferentes gerenciadores de trilhas. Por isso, o eProfile oferece também suporte à comunicação com vários gerenciadores de trilhas, segundo exemplificado na Figura 11.

4.2 Arquitetura

A Figura 12 apresenta a arquitetura do eProfile. Os componentes “Aplicação” e “Gerenciador de Trilhas” são os mesmos mostrados na Figura 10, enquanto o componente “eProfile” foi expandido para mostrar seu funcionamento interno.

Cada aplicação tem a possibilidade de montar seu próprio modelo de entidade. Para que o eProfile processe os perfis baseados em um modelo de entidade, ele precisa de três informações: (i) a trilha da entidade, (ii) o modelo de entidade utilizado pela aplicação e (iii) as regras

Figura 12: Arquitetura do eProfile



Fonte: elaborado pelo autor

de inferência para o perfil. Isto é visível na Figura 12, onde o agente "Raciocinador" recebe dados destes três bancos. A trilha da entidade é carregada pelo eProfile a partir do gerenciador de trilhas, enquanto as outras duas informações são enviadas diretamente da aplicação para o eProfile. Desta forma, as regras de inferência definidas pela aplicação serão utilizadas para inferir o perfil da entidade.

As setas da Figura 12 indicam a direção do fluxo dos dados, onde podem se verificar dois pontos de entrada e um ponto de saída.

O primeiro ponto de entrada são as informações enviadas pela aplicação ao eProfile ou, mais especificamente, ao agente **configurador**. As informações que devem ser enviadas pela aplicação são as seguintes:

- o modelo de entidade utilizado pela aplicação, ou seja, o formato do perfil da entidade. Uma vez que a trilha da entidade seja processada, as informações inferidas serão armazenadas e disponibilizadas para a aplicação neste formato;

- as regras para inferência do perfil. Estas regras serão executadas em períodos regulares, de modo a atualizar o perfil da entidade (ver Seção 4.5). O configurador tem a função de receber as regras, validá-las e inseri-las na **base de regras**. Juntamente com as regras, é especificado a criticidade da informação. O sistema utilizará esta informação para determinar com que frequência a inferência desta informação será executada.

O segundo ponto de entrada é aquele através do qual são recebidas as trilhas das entidades, que são buscadas do gerenciador de trilhas pelo eProfile. Como o eProfile pode se conectar a diferentes gerenciadores de trilha (Figura 11), onde em cada um deles a trilha tem um formato específico, é necessário que seja realizada uma conversão entre o formato original e o formato utilizado pelo eProfile, para que as trilhas sejam armazenadas de forma padronizada. Este trabalho é realizado por um agente intitulado **conversor**, que é capaz de realizar a conversão para os formatos de trilha suportados pelo eProfile.

A trilha da entidade é então armazenada no eProfile em uma ontologia de trilha, chamada **eOntoTrail**. Esta ontologia possui as classes básicas para a construção de uma trilha, sendo que a mesma pode ser estendida de acordo com a especificação do gerenciador de trilhas que está sendo utilizado. A forma como ocorre esta extensão é detalhada na Seção 4.4.1.

Uma vez de posse das trilhas, do modelo e das regras, o agente **raciocinador** (Seção 4.3.2) realiza a inferência na trilha, gerando assim um perfil da entidade, que será uma extensão de uma ontologia de perfil básica denominada **eOntoProfile** (ver Seção 4.4.2). Esta ontologia, da mesma forma que a ontologia de trilha, contém classes básicas a partir das quais as aplicações vão especializar, utilizando-as para armazenar informações próprias. A aplicação pode então conectar-se ao eProfile e buscar o perfil de sua entidade, cujos dados são enviados pelo agente **comunicador**.

4.3 Modelagem dos Agentes

Como o eProfile é um modelo que infere informações de forma autônoma a respeito de dados armazenados em uma ontologia, e está organizado em partes independentes, a tecnologia de agentes autônomos foi escolhida para o desenvolvimento. A metodologia utilizada para a modelagem é a Prometheus (PADGHAM; WINIKOFF, 2002). Esta metodologia define um processo detalhado para a especificação, projeto, implementação e teste/depuração de sistemas de software orientados a agentes (PADGHAM; WINIKOFF, 2004).

4.3.1 Especificação do Modelo

A metodologia Prometheus define que o ponto central da modelagem de um sistema é a identificação dos objetivos iniciais do sistema. De acordo com a Seção 4.1, o eProfile é um modelo que gerencia perfis de entidades, permitindo que as aplicações definam (i) seus próprios modelos de entidade e (ii) regras através das quais a trilha de uma entidade será processada. Um processo de inferência baseado nestas informações monta ou atualiza um perfil de entidade, que é então disponibilizado para a aplicação.

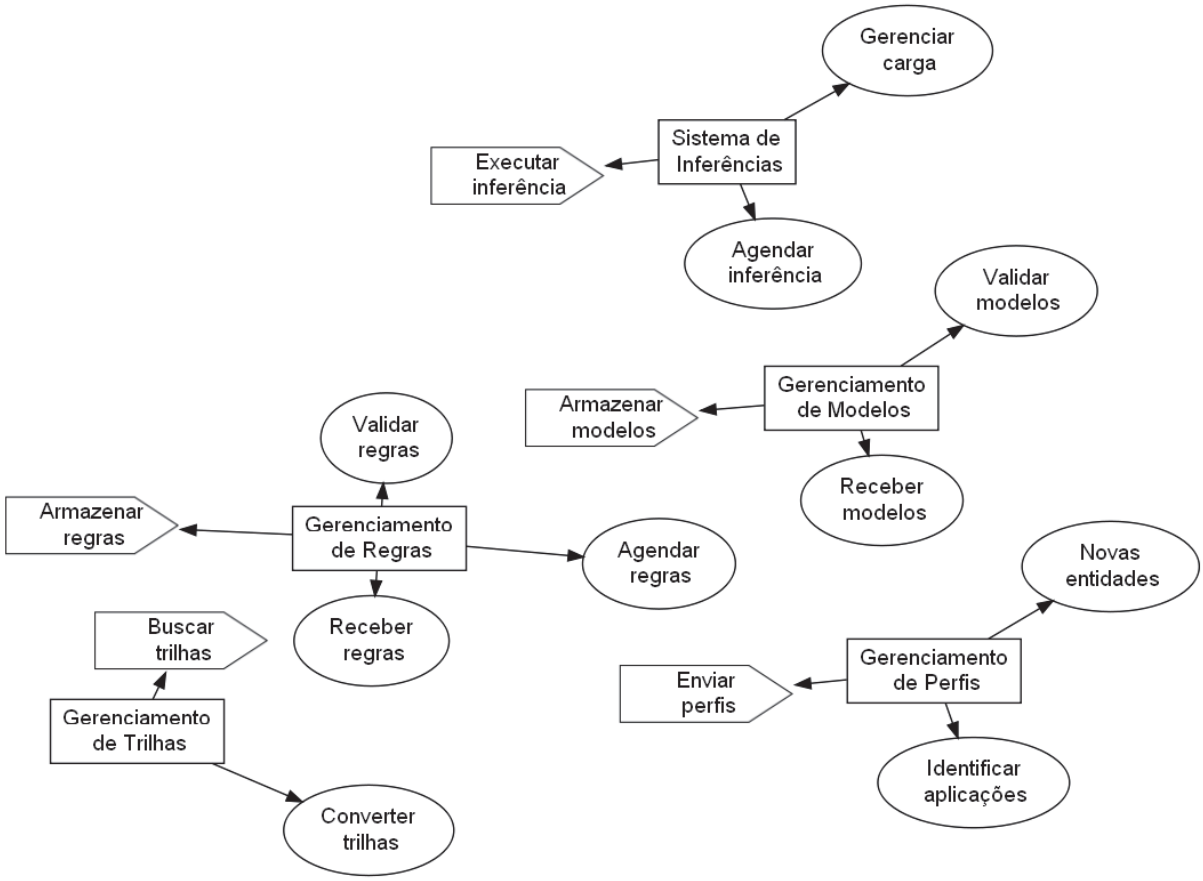
A partir desta descrição, identificam-se os objetivos. Os respectivos subobjetivos são definidos considerando a forma como cada objetivo pode ser atingido (PADGHAM; WINIKOFF, 2004). Assim, os objetivos e subobjetivos do eProfile são:

- gerenciar múltiplos perfis de entidade
  - permitir que aplicações se identifiquem;
  - aceitar requisições de novas entidades;
  - disponibilizar os perfis inferidos à aplicação;
- permitir que as aplicações definam seus próprios modelos de entidade
  - receber novos modelos de entidade da aplicação;
  - validar os modelos de entidade;
  - armazenar os modelos de entidade;
- permitir que as aplicações definam suas regras de inferência
  - receber as regras;
  - validar as regras;
  - armazenar as regras;
  - gerenciar o agendamento da execução das regras;
- buscar as trilhas de uma entidade a partir de um gerenciador de trilhas
  - interagir com os gerenciadores de trilhas suportados;
  - converter as informações do formato de trilha original para a ontologia de trilhas do eProfile;
- inferir a trilha, montando um perfil de entidade;
  - agendar a execução das regras;
  - executar a inferência de acordo com as regras;
  - gerenciar sobrecarga do servidor e reagendar as regras, se necessário.

Para que a especificação do modelo seja completa, a metodologia propõe que sejam identificadas as *interfaces*, ou seja, os pontos onde o modelo se comunica com outros sistemas. Os pontos onde o modelo recebe informações de outros sistemas são chamados "preceitos", e os pontos onde o modelo envia informações para outros sistemas são chamados "ações".

A partir dos subobjetivos, é possível extrair as funcionalidades do modelo. A Figura 13 identifica as funcionalidades do eProfile, com as caixas representando as funcionalidades (baseadas nos objetivos), as formas ovais representando os subobjetivos, e os ícones pentagonais representando as ações. Cada objetivo possui pelo menos uma ação que é executada através de um gatilho, e utiliza um conjunto de informações, de acordo com a Tabela 2.

Figura 13: Funcionalidades do eProfile



Fonte: Elaborado pelo autor

Os preceitos do modelo, isto é, os pontos onde o eProfile recebe informações de outros sistemas são: (i) registro de nova aplicação, (ii) registro de nova entidade, (iii) modelo definido pela aplicação, (iv) regras para inferência das trilhas e (v) as próprias trilhas. O eProfile possui apenas uma ação, isto é, apenas uma saída de informações, que são os perfis das entidades que são gerados pelo modelo.

É necessário também a identificação das bases de dados necessárias para o funcionamento do modelo. O eProfile faz uso das seguintes bases de dados:

- Base de dados de aplicações: contém a lista de aplicações registradas no eProfile;
- Base de dados de entidades: contém a listagem de entidades identificadas pelas aplicações;
- Base de dados de modelos de entidades: contém os modelos de entidade registrados pela aplicação;
- Base de dados de regras de inferência: contém as regras de inferência que foram enviadas pela aplicação;

Tabela 2: Funcionalidades do eProfile

Funcionalidade	Ação	Gatilho	Informações usadas	Informações geradas
Gerenciamento de perfis	Enviar perfis	Requisição da aplicação	Base de perfis	Perfil
Gerenciamento de regras	Armazenar regras	Envio pela aplicação	Regra	Base de regras
Gerenciamento de modelos	Armazenar modelos	Envio pela aplicação	Modelo	Base de modelos
Gerenciamento de trilhas	Buscar trilhas	Agendamento	Nenhum	Base de trilhas
Sistema de inferência	Executar inferência	Agendamento	Base de regras, modelos e trilhas	Base de perfis

Fonte: elaborado pelo autor

- Base de dados de trilhas: contém as trilhas buscadas do gerenciador de trilhas, e convertidas para o formato do eProfile;
- Base de dados de perfis de entidades: contém os perfis de entidades, já inferidos a partir dos modelos, regras e trilhas.

As bases de dados estão presentes nas Figuras 12, 14 e 15, embora no primeiro caso apenas as mais importantes são mostradas.

4.3.2 Arquitetura dos Agentes

A Figura 14 apresenta o diagrama de ligação (*coupling*) do eProfile. A partir deste diagrama, é possível agrupar as bases de dados e funcionalidades em grupos afins, e identificar os agentes resultantes, como é mostrado na Figura 15. Quatro agentes foram identificados: (i) comunicador, (ii) configurador, (iii) conversor e (iv) raciocinador.

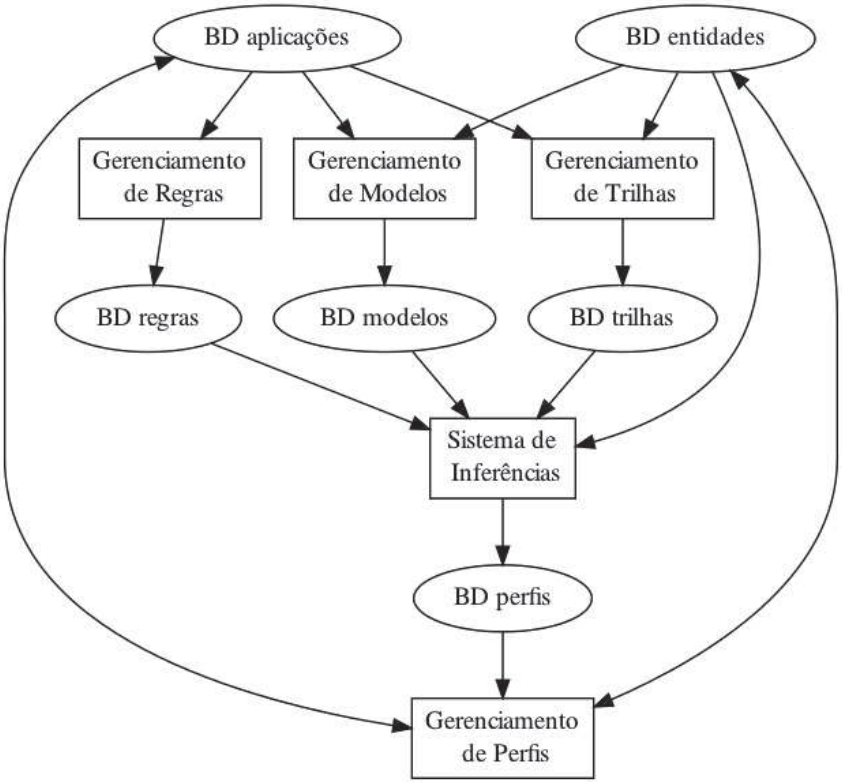
O **agente comunicador** é responsável pela comunicação entre as aplicações e o eProfile. Ele cadastra as aplicações na base de dados de aplicações e permite que as aplicações cadastrem as entidades na base de dados de entidades. Ele também disponibiliza os perfis inferidos pelo agente raciocinador para que sejam recuperados pelas aplicações.

O **agente configurador** recebe as seguintes configurações das aplicações: modelos de entidade e regras. Quando um modelo de entidade é recebido, ele é validado e, se aceito, é armazenado na base de dados de modelos de entidades.

O agente realiza também o recebimento das regras enviadas pela aplicação. Uma vez que as regras sejam recebidas, elas são validadas e, se forem consideradas válidas, são armazenadas na base de regras. A aplicação é então informada do aceite ou rejeição. O agente pode rejeitar uma regra se a mesma for inválida.

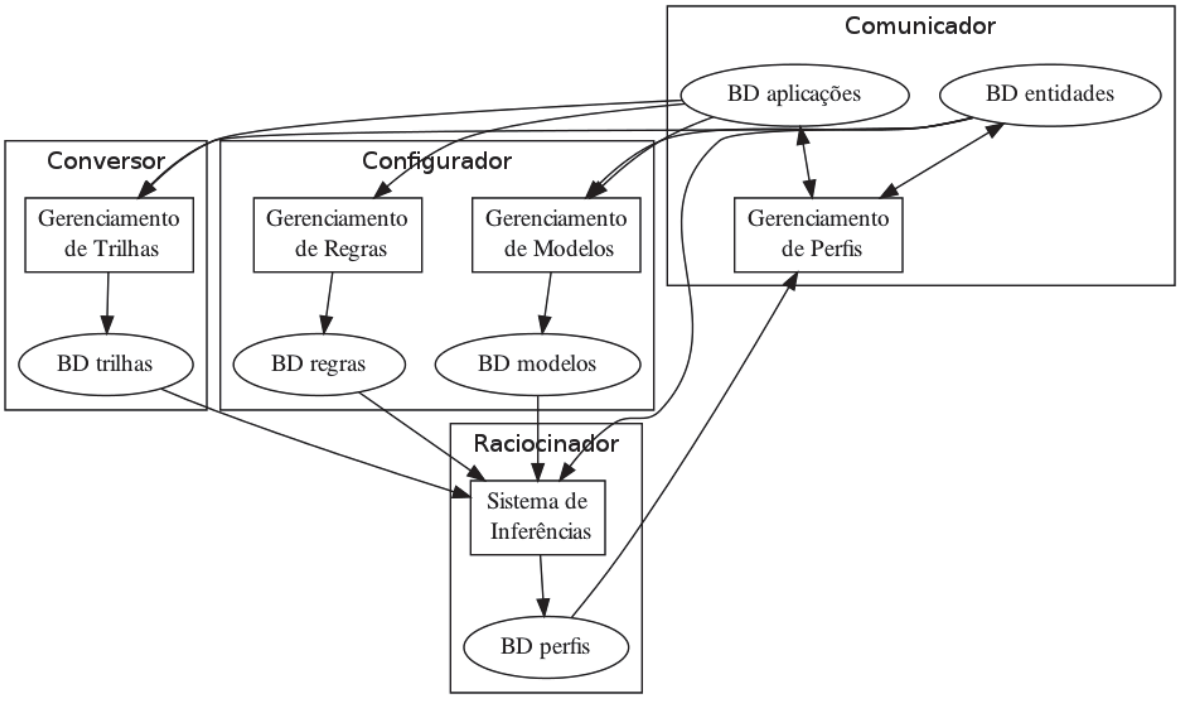
O **agente conversor** realiza a conversão dos dados da trilha do gerenciador de trilhas (externo) para a ontologia de trilhas do eProfile (interno). A trilha do gerenciador pode ser dispo-

Figura 14: Diagrama de ligação do eProfile



Fonte: Elaborado pelo autor

Figura 15: Diagrama de ligação do eProfile, com a identificação dos agentes



Fonte: Elaborado pelo autor

nibilizada em qualquer formato e por qualquer meio, desde que o eProfile ofereça suporte ao gerenciador de trilhas específico. O agente conversor tem também a responsabilidade de verificar, a cada período pré-determinado, atualizações nos dados disponibilizados no gerenciador.

O **agente raciocinador** infere nas trilhas de acordo com as regras enviadas pela aplicação, e atualiza o perfil da entidade. A execução do agente não necessita que o aplicativo ou o gerenciador de trilhas esteja conectado no momento da execução da inferência, e ocorre da seguinte forma:

1. o agente verifica as regras, de acordo com a solicitação feita pelo agente configurador;
2. a regra é executada através de um raciocinador;
3. caso a regra tenha sido executada com sucesso, o resultado da regra é atualizado no perfil. Em caso de insucesso, o erro é enviado para a aplicação. Se a aplicação não estiver conectada ao eProfile, o resultado é enviado à aplicação no momento que a mesma se reconectar.

A Figura 16 mostra o diagrama da visão geral do modelo, que junta todas as partes da arquitetura. Esta figura é comparável à Figura 12, porém apresentando uma representação mais completa.

4.4 Ontologias

O eProfile utiliza duas ontologias: uma ontologia de trilhas e uma ontologia de perfis. As ontologias utilizadas pelo eProfile são estendidas através de subclasses de acordo com o formato da trilha enviada pelo gerenciador (no caso da ontologia de trilha) ou das necessidades da aplicação (no caso da ontologia do perfil). Isto é exemplificado na Figura 17, onde o eProfile oferece uma classe genérica “Atividade”, que é especializada pela classe “AtividadeDeCompra”, utilizada por um dos gerenciadores de trilha.

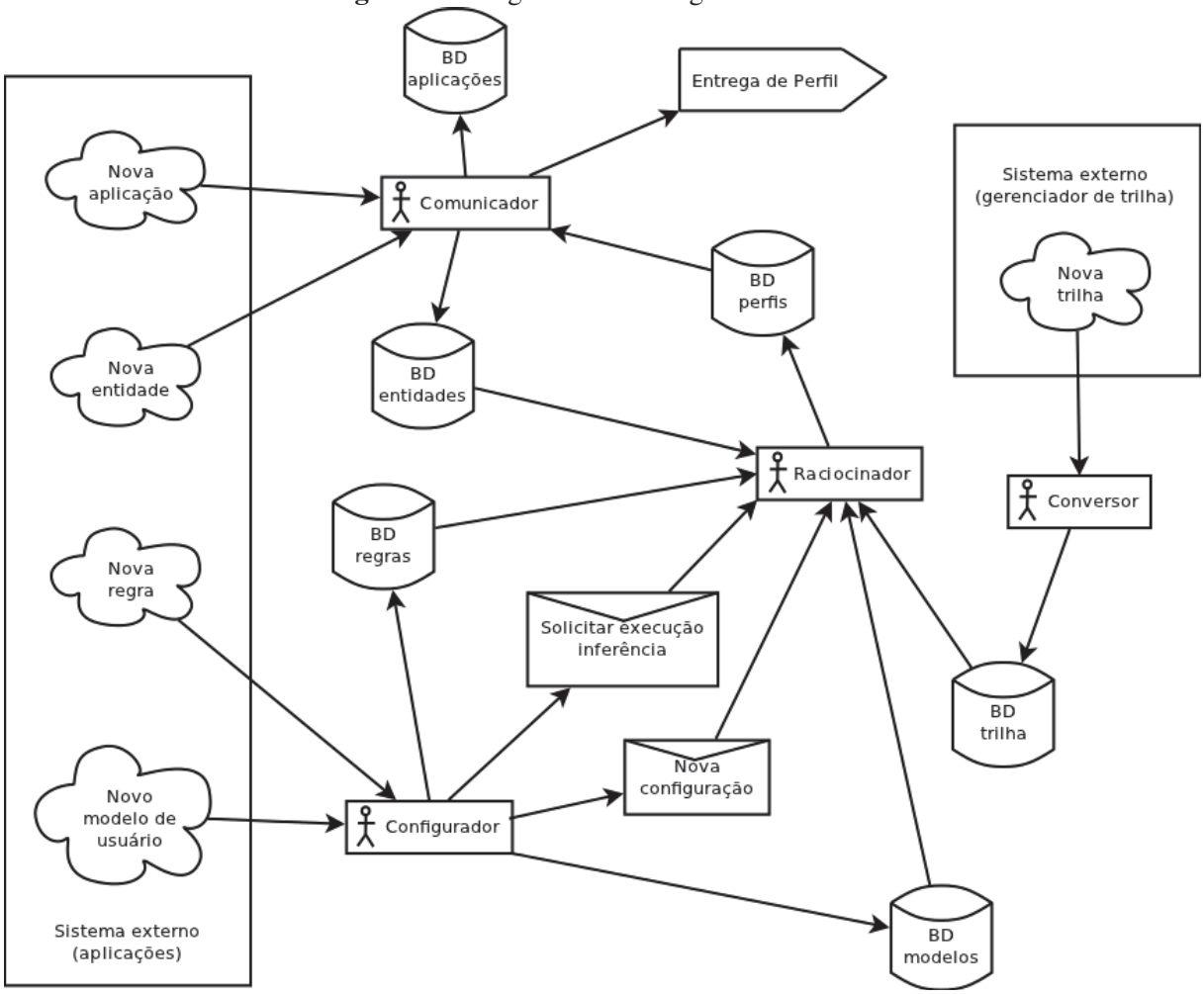
4.4.1 Ontologia de Trilha (eOntoTrail)

O eProfile implementa uma ontologia de trilha, chamada **eOntoTrail**. Esta ontologia será alimentada a partir dos diferentes gerenciadores de trilhas suportados, onde o agente conversor é responsável por receber os dados no formato do gerenciador de trilhas e convertê-los para a eOntoTrail. Assim, o raciocinador terá uma ontologia única sobre a qual poderá raciocinar.

A eOntoTrail está representada na Figura 18, onde todos os relacionamentos, representados pelas setas, são do tipo ou “has-a” ou “has-some”.

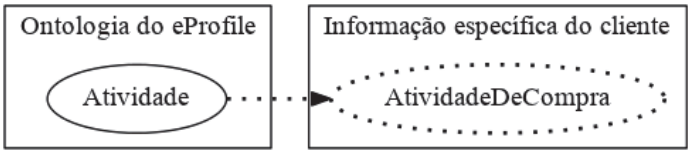
Abowd et al. (1999) definem que aplicações sensíveis a contexto estão interessadas em saber **quem**, **onde**, **quando** e **o quê** (isto é, o que o usuário está fazendo), de forma a poderem descobrir por que a situação está ocorrendo. Assim, uma trilha é representada por um conjunto de eventos caracterizados pelas seguintes classes:

Figura 16: Diagrama da visão geral do modelo



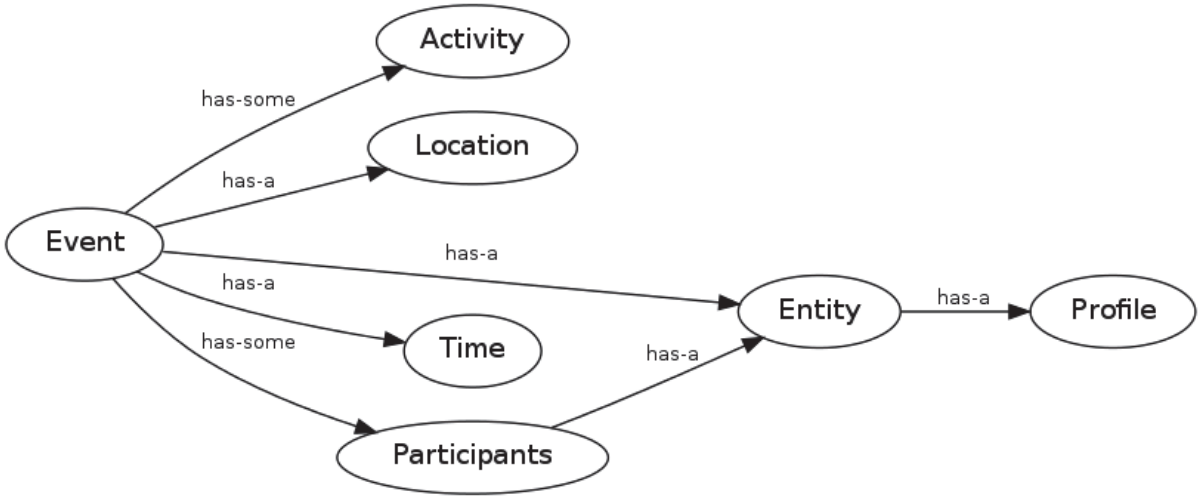
Fonte: Elaborado pelo autor

Figura 17: Exemplo da extensão de uma classe do eProfile



Fonte: elaborado pelo autor

Figura 18: Ontologia de Trilha do eProfile

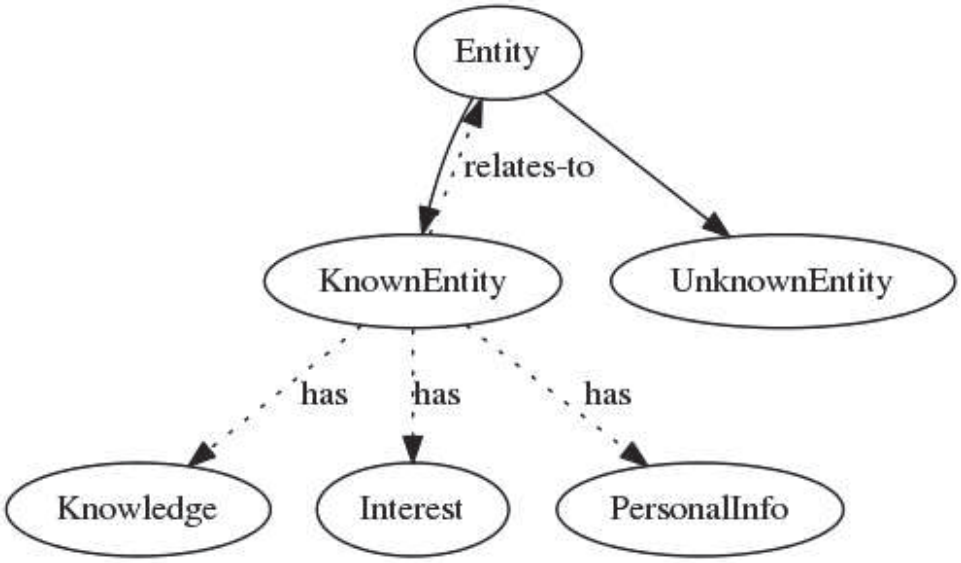


Fonte: elaborado pelo autor

- *Entity:* é a entidade que está realizando a ação. Esta entidade é vinculada à ontologia descrita na Seção 4.4.2;
- *Participants:* uma lista de outras entidades relacionadas à ação. No caso de uma reunião de negócios, por exemplo, o perfil que está sendo analisado é do tipo *Entity*, enquanto os outros participantes são membros da classe *Participants*;
- *Location:* a localização onde está ocorrendo a ação. Uma localização pode ser uma localização física específica (como uma sala) ou não (uma estrada, por exemplo). A localização pode estar vinculada a uma região que, opcionalmente, pode estar vinculada a outras subregiões e geralmente possui uma localização global (GPS) vinculada a ela. Outros tipos de localização podem ser estendidas a partir da classe;
- *Time:* data e hora em que ocorreu o evento;
- *Activity:* refere-se à ação que a entidade está executando (ou tentou executar). É a classe mais abrangente da ontologia, e a que possivelmente será a mais especializada pelos gerenciadores de trilha.

Juntamente com as informações acima, as aplicações podem armazenar **informações contextuais** na trilha. Informações contextuais são informações que classificam ou expandem classes utilizadas na ontologia. Se, por exemplo, uma ontologia faz referência a uma determinada sala denominada `SalaReuniao`, e esta sala é registrada como uma entidade na ontologia, a mesma será uma classe derivada da classe `Entity`. No entanto, havendo um classe `Sala` definida nas informações contextualizadas desta trilha, a classe `SalaReuniao` pode também ser derivada da classe `Sala`, fornecendo assim mais informações sobre a ontologia. Um exemplo mais completo deste tipo de informação contextualizada pode ser verificado na Seção 4.6.

Figura 19: Ontologia de Perfil do eProfile



Fonte: elaborado pelo autor

4.4.2 Ontologia de Perfil (eOntoProfile)

A ontologia de perfil é a ontologia utilizada para armazenar os modelos de entidade, bem como as informações de perfil que são inferidas a partir da trilha. Para que a ontologia seja viável dentro do modelo proposto, ela deve ser uma ontologia genérica que permita que suas classes sejam estendidas pelas aplicações.

A ontologia proposta, a qual chama-se eOntoProfile, está representada na Figura 19. Nesta figura, um relacionamento entre as classes é mostrado através de uma linha pontilhada, e uma especialização de uma classe é mostrado por uma linha contínua. A eOntoProfile é uma ontologia que será estendida pelas aplicações de modo a montarem seus próprios modelos de entidade.

A classe básica da ontologia é a classe *Entity*. A classe *Entity* representa qualquer entidade no mundo real. Ela tem como filhos as classes:

- *KnownEntity:* representa uma entidade conhecida, isto é, uma entidade que tenha um perfil associado a ela. Por exemplo, se uma pessoa tem um perfil no eProfile e é registrado um evento de trilha associado a outra pessoa **que também possua um perfil**, esta segunda pessoa é registrada como uma *KnownEntity* neste evento;

- *UnknownEntity:* representa uma entidade que não possui um perfil associado a ela. Por exemplo, uma trilha registra o evento em que uma pessoa entra em uma loja, mas a loja não tem um perfil no eProfile. Neste caso, a loja é uma *UnkownEntity*;

Esta diferenciação é importante porque no primeiro caso, o eProfile tem acesso a uma vasta gama de informações a respeito da entidade, das entidades relacionadas àquela entidade, e assim por diante. No caso de uma *UnknownEntity*, o perfil não está disponível, e é apenas um nome para o eProfile.

A ontologia FOAF (BRICKLEY; MILLER, 2010) utiliza o conceito de *knows* (conhece) para vincular duas pessoas. A eOntoProfile estende este conceito através da propriedade *relates-to* (se-relaciona-com) que vincula dois elementos. Através desta propriedade, é possível não apenas vincular os elementos, mas também descrever o tipo de relação que eles tem. Exemplos de relações são *friend-of* (amigo-de), representando uma amizade entre duas pessoas, *employee-of* (empregado-de), representando um vínculo empregatício entre duas pessoas, *owns* (possui), representando que uma pessoa possui um dispositivo, *depends* (depende), onde um dispositivo depende de outro, e qualquer outra relação que a aplicação deseje criar.

Uma entidade pode conter diversas características. As características das entidades são inspiradas no modelo proposto por Golemati et al. (2007), embora o modelo tenha sido simplificado:

- *PersonalInfo:* contém quaisquer informações próprias da entidade, tais como nome, localização atual, características, entre outros;
- *Interest:* quaisquer interesses ou preferências que a entidade tenha, como preferência musical ou preferência por um tema de cores específico em suas aplicações;
- *Knowledge:* conhecimento possuído pela entidade. Pode referir-se a conhecimento formal (tal como formação acadêmica) ou informal (habilidade ou experiência em alguma atividade).

Os modelos de entidade estenderão estas categorias de acordo com as suas necessidades.

Todos os modelos de entidade definidos para um perfil serão estendidos em uma mesma ontologia. Isto possibilitará a interpersonalização, ou seja, permitirá com que os dados gerados por uma aplicação possam ser utilizadas por outra. Assim, existe a possibilidade de conflito de nomes, ou seja, uma aplicação pode criar uma classe chamada *Preference* onde ela registra preferências da entidade por um determinado tipo de livros, enquanto outro aplicativo cria uma classe com o mesmo nome para registrar as preferências de tela da aplicação.

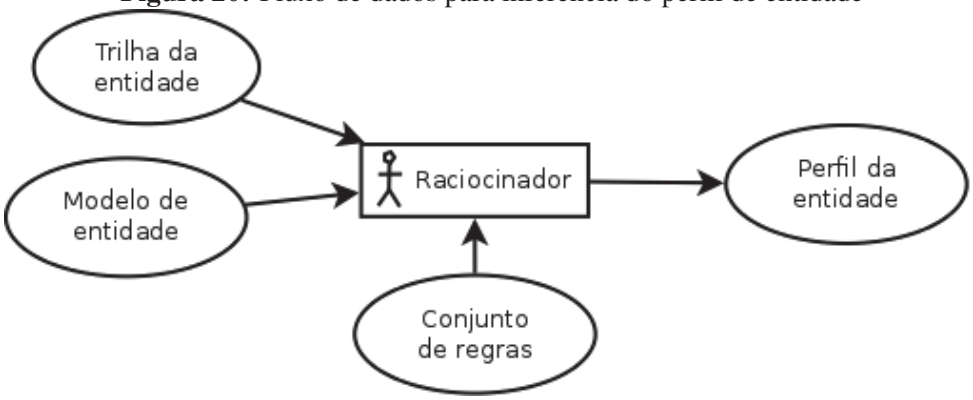
Para evitar este conflito, um nome único de aplicação deve ser concatenado ao nome da classe. Assim, se um dos aplicativos utilizasse o nome *books* e o outro utilizasse o nome *gps*, teriam-se, respectivamente, as classes *books.Preference* e *gps.Preference*.

Para que um aplicativo possa ter acesso aos dados gerados por outro, existem duas possibilidades:

1. um aplicativo acessar diretamente os dados gerados pelo outro. No exemplo acima, o aplicativo *books* acessaria diretamente os dados da classe *gps.Preference*;
2. os aplicativos utilizarem um contrato onde eles concordam em utilizar os mesmos campos. Assim, *books* e *gps* poderiam utilizar o nome *findlibrary*, criando o campo *findlibrary.Preference* onde ambos gravariam e leriam as informações.

Todos as extensões (subclasses) da ontologia são definidas pelas aplicações, e todas as informações do perfil são inferidas. A forma como esta inferência ocorre é descrita na Seção 4.5.

Figura 20: Fluxo de dados para inferência do perfil de entidade



Fonte: elaborado pelo autor

4.5 Regras e Inferência

A aplicação de regras é peça fundamental para o funcionamento do eProfile. Uma regra é uma instrução de inferência registrada por uma aplicação que será executada sobre trilhas de uma entidade, resultando num perfil de entidade de acordo com um modelo de entidade previamente cadastrado pela aplicação.

Para que uma regra possa ser aplicada, as seguintes informações precisam ter sido previamente registradas no eProfile:

- a regra propriamente dita;
- um modelo de entidade, que servirá como base para o formato de perfil da entidade;
- a trilha da entidade, sobre a qual a inferência será executada.

A partir destas três informações, o raciocinador tem condições de gerar um perfil de entidade, de acordo com a Figura 20.

As regras são sempre definidas pela aplicação, e uma regra é composta pelo seguinte conjunto de informações:

- **Nome:** um identificador único para a regra, que permite que a aplicação identifique a regra posteriormente, para que possa editá-la ou excluí-la;
- **Campo do perfil:** define qual é o campo do perfil da entidade que deverá ser atualizado através da execução desta inferência. Este campo deve ser obrigatoriamente equivalente a uma classe definida na ontologia de modelo de entidade;
- **Regra:** a regra que será executada sobre a trilha da entidade;
- **Atualização:** uma instrução que define como o campo será atualizado (por exemplo, o valor pode ser simplesmente sobrescrito ou atualizado de acordo com alguma fórmula);

- **Frequência de atualização:** define a periodicidade de execução da regra. É importante notar que o valor definido aqui pode não ser honrado, devido a outros motivos (como uma sobrecarga no servidor);

- **Validade**: define qual a data limite para a execução desta regra. A ausência desta informação define que a regra continuará sendo processada indefinidamente.

O raciocinador gerenciará a carga de processamento no servidor. Assim, algumas regras podem não ser executadas de acordo com o agendamento solicitado. Além disso, é possível que uma aplicação registre certas regras mas deixe de usar o servidor, fazendo com que as regras continuem sendo executadas desnecessariamente. Assim, caso um perfil não seja consultado por um tempo determinado, a frequência da execução das regras passa a diminuir progressivamente, até o ponto em que a regra não seja mais executada. Caso a aplicação volte a consultar os perfis, a frequência de execução das regras é normalizada.

As regras podem ser escritas em duas linguagens diferentes: SPARQL ou SWRL. O SPARQL (W3C, 2008) é uma linguagem de consulta de ontologias cuja sintaxe assemelha-se a linguagem de consulta de banco de dados SQL (ISO/EIC, 1992), enquanto o SWRL (W3C, 2004) é uma linguagem de regras que permite a consulta utilizando construtores lógicos mais complexos que o SPARQL. Além disso, as informações podem ser inferidas através das relações axiomáticas entre as classes.

Uma vez que a regra tenha sido validada pelo agente configurador, ela entra em funcionamento e passa a ser elegível para execução.

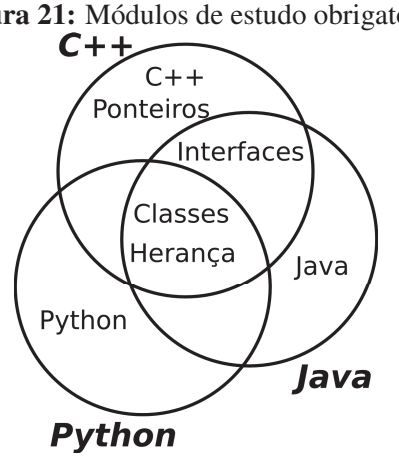
Tanto as consultas via SPARQL/SWRL quanto a geração de classes a partir das relações axiomáticas são executadas através de um motor de inferências externo que é descrito na Seção 5.1.1. Um exemplo detalhado da aplicação de regras pode ser verificado na Seção 4.6.

4.6 Exemplo de operação

Nesta seção é mostrado um exemplo de funcionamento do eProfile, detalhado nas seções anteriores. O objetivo do exemplo é demonstrar o processo através do qual um campo de perfil de uma entidade é atualizado.

Este exemplo mostra um curso de programação em linguagens orientadas a objeto, na qual os estudantes podem escolher quais assuntos específicos dentro da matérias eles vão estudar. O curso consiste em um conjunto de módulos relacionados a orientação a objetos que o usuário transfere para seu dispositivo móvel e estuda através dele. Destes módulos fazem parte conceitos básicos da matéria (classes, herança, *interfaces* e ponteiros) e linguagens de programação orientadas a objeto (C++, Python e Java). O estudante não é obrigado a concluir todos os módulos. No final do curso, no entanto, ele deve escolher a linguagem de programação que ele utilizará para responder as questões da prova final. Dependendo da linguagem que escolher, ele deve ter concluído, no mínimo, os módulos referentes àquela linguagem.

Figura 21: Módulos de estudo obrigatórios



Fonte: Elaborado pelo autor

As dependências dos módulos estão representadas na Figura 21. Por exemplo, caso o aluno decida realizar a prova em Java, ele deve ter estudado classes, herança e *interfaces*, além do Java propriamente dito. Como a linguagem Java não contém o conceito de ponteiros, ele não é obrigado a ter cursado este módulo.

Neste exemplo, um gerenciador de conteúdo distribui os módulos aos alunos, que o acessam através de seus dispositivos pessoais. O gerenciador de conteúdo registra em um gerenciador de trilhas cada vez que um módulo é iniciado ou concluído. O gerenciador de conteúdo, o gerenciador de trilhas e o eProfile conectam-se de acordo com a Figura 10 (Seção 4.1).

Cada evento (início ou conclusão de um módulo) é registrado contendo as seguintes informações: (i) entidade que realizou o evento, (ii) módulo, (iii) tipo de evento (início ou conclusão), (iii) data e hora e (iv) se é a primeira vez que o estudante realiza este tipo de evento. A definição da primeira vez que o estudante realiza o módulo é importante, pois ele pode realizar o mesmo módulo várias vezes, mas apenas a primeira vez é utilizada na contagem dos módulos realizados.

Neste exemplo, os eventos descritos na Tabela 3 foram registrados pelo aplicativo no gerenciador de trilhas durante um período de um dia. A Entidade é derivada da classe “Entity”, a data e hora são derivadas da classe “Time”, e as outras informações são derivadas da classe “Activity”, de acordo com a ontologia definida na Figura 18.

No exemplo apresentado, deseja-se atualizar o perfil de uma entidade, inserindo a informação de proficiência dela em determinado assunto. Esta informação será utilizada posteriormente pela aplicação para determinar se o estudante está apto a realizar o teste. O perfil da entidade (aluno) que será gerado é simples para este exemplo, possuindo apenas um campo: proficiência em Java.

Para que o eProfile possa ser capaz de inferir esta informação, a aplicação precisa, nas trilhas, enviar informações contextuais suficientes para que o eProfile possa montar a ontologia que represente a situação atual. Um exemplo de informação deste tipo são as dependências entre as disciplinas. Para que o eProfile possa ser capaz de compreender o contexto dos módulos de

Tabela 3: Trilha gerada pelo aplicativo

Entidade	Objeto de aprendizagem	Tipo de evento	Data e Hora	Primeira vez?
Andre	Interfaces	Início	09/07/2012 08:12	Sim
Andre	Classes	Término	09/07/2012 09:28	Sim
Maria	Herança	Término	09/07/2012 09:32	Sim
Andre	Java	Término	09/07/2012 11:14	Sim
Maria	Interfaces	Término	09/07/2012 11:28	Sim
Andre	Ponteiros	Início	09/07/2012 13:58	Sim
Andre	Ponteiros	Término	09/07/2012 14:12	Sim
Andre	Java	Término	09/07/2012 16:18	Não
Andre	Interfaces	Término	09/07/2012 17:32	Sim

Fonte: Elaborado pelo autor

estudo e dos assuntos, é necessário que a aplicação (neste caso, o gerenciador de conteúdo) envie a informação contextual relevante, que está representada na Figura 22.

A informação contextual pode ser enviada através da trilha apenas uma vez (caso ela seja reenviada, a nova informação sobrescreve a anterior). Esta informação é recebida pelo eProfile através do módulo configurador e armazenada na base de dados de trilhas. A partir deste momento, neste exemplo, o eProfile poderá identificar uma referência à "Java" ou "Python", identificando-as como Linguagens de Programação.

Além das trilhas, a aplicação deve enviar as regras de inferência ao eProfile. Estas regras serão utilizadas para inferir as trilhas e gerar o perfil. Para que o eProfile possa identificar se uma entidade é proficiente em alguma disciplina, são necessárias três regras: (i) a identificação dos módulos requeridos para a proficiência na disciplina; (ii) a identificação dos módulos concluídos pelo estudante e; (iii) a verificação se os objetos requeridos foram concluídos.

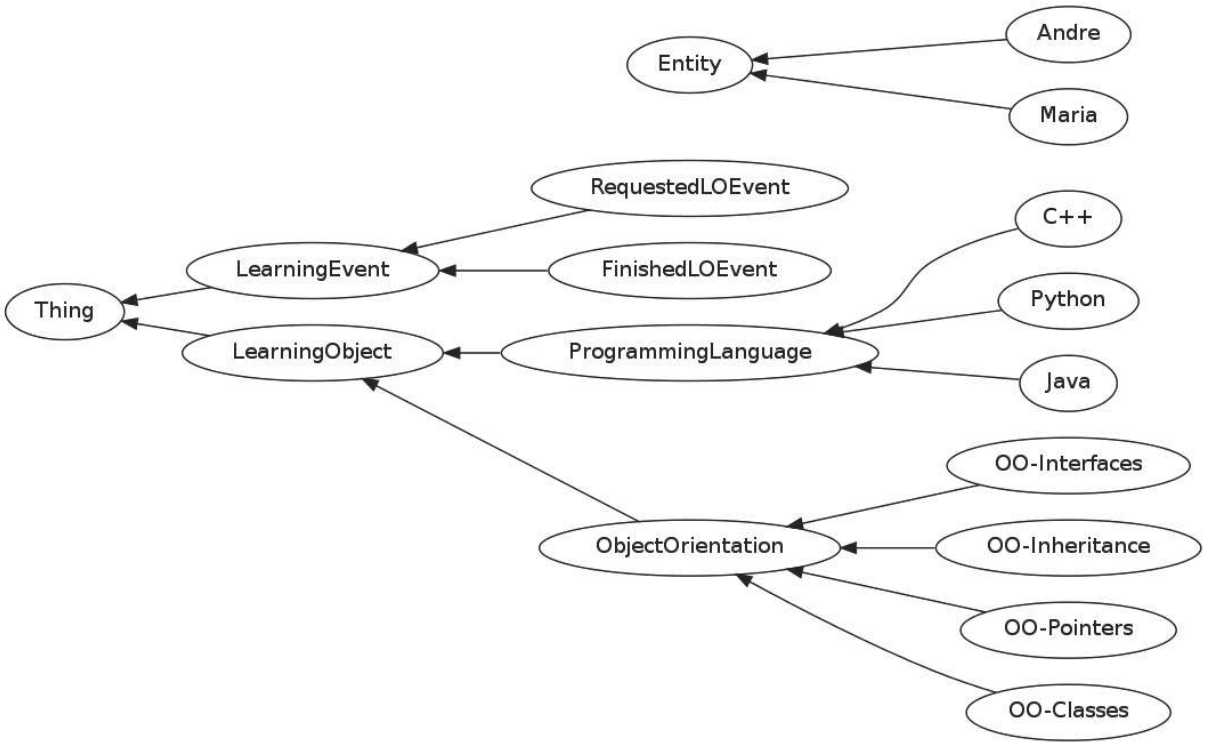
A primeira regra está apresentada na Figura 23, sendo composta por uma classe escrita em OWL[1]. Para que um estudante seja considerado proficiente em Java, ele deve ter concluído os módulos que fazem relação com a linguagem Java (neste caso classes, herança e *interfaces*, na linha 4), bem como o módulo da própria linguagem Java (linha 3).

O segundo passo é a identificação dos módulos concluídos pelo estudante relacionados ao Java. Esta regra é apresentada na Figura 24. Assim, são considerados apenas os eventos de conclusão de objeto de aprendizagem (linha 3) realizados pela entidade (linha 4), relacionados ao Java (linha 5) e que tenham sido realizados pela primeira vez (linha 7), uma vez que o aluno pode estudar o mesmo material várias vezes. O parâmetro `[?entity]` é substituído pela entidade na execução da regra de cada perfil. É importante notar que para verificar se a disciplina está relacionada ao Java, a regra criada na Figura 23 é utilizada (linhas 5 e 6).

Uma vez que estas duas regras estejam armazenadas na base de dados de regras, o eProfile executa o raciocinador, que gera uma nova ontologia OWL, com as classes inferidas identifica-

[1]Embora o eProfile utilize internamente a sintaxe OWL/RDF (W3C, 2011b), a sintaxe Manchester-OWL está sendo usada neste exemplo por ser mais compacta e amigável (W3C, 2009).

Figura 22: Informações contextuais do cenário apresentado



Fonte: elaborado pelo autor

Figura 23: Regra de requerimentos para disciplina

```
Class: <el:JavaRequirement>
    EquivalentTo:
        <el:Java>
         or (<el:relatesTo> some <el:Java>)
```

Fonte: elaborado pelo autor

Figura 24: Regra de módulos concluídos

```
Class: <el:JavaLessonsTaken>
    EquivalentTo:
        <el:FinishedLOEvent>
         and (<el:hasEntity> some [?entity])
         and (<el:hasLearningObject> some
                              <el:JavaRequirement>)
         and (<el:isFirstTime> value true)
```

Fonte: elaborado pelo autor

Figura 25: Regra de proficiência

```
1 JavaProficience <=
2   (Count(SubClasses(JavaRequirement))) equals
3           (Count(SubClasses(JavaLessonsTaken)))
```

Fonte: elaborado pelo autor

Figura 26: Regra para processamento do perfil do estudante

```
1 Entity:    Student
2 Field:     Knowledge\JavaProficience
3 Rule:      JavaProficience
4 Update:    update
5 Frequence: 300
6 Expires:   never
```

Fonte: elaborado pelo autor

das como filhas das classes criadas.

A terceira regra é uma regra de atualização de perfil (Figura 25). Nesta regra, são comparados o número de módulos relacionados ao Java (regra da Figura 23) com o número de módulos relacionados ao Java que foram concluídos pela entidade (regra da Figura 24). Se o número for igual, o usuário é considerado proficiente, e esta regra é atualizada no campo de proficiência no perfil da entidade.

A Figura 26 mostra a definição da regra. A definição informa o eProfile que o campo JavaProficience (linha 2) das entidades do tipo Estudante (linha 1) será atualizado com o resultado da regra chamada JavaProficience (linha 3), que é a regra mostrada na Figura 25. O prefixo "Knowledge" da linha 2 indica que este campo é filho da classe *Knowledge* na ontologia de perfil apresentada na Figura 19. A regra de atualização da linha 4 indica que, cada vez que a regra for executada, o valor resultante deve substituir o valor anterior. A regra será executada a cada 300 segundos (linha 5) e não expira (linha 6).

O perfil gerado pela aplicação da regra é uma ontologia OWL, mostrada na Figura 27. A ontologia contém as classes Andre e Maria (linhas 13 e 17), que representam as duas entidades deste exemplo. Pode-se verificar no perfil que a entidade Andre possui na propriedade *hasKnowledge* a classe *JavaProficience*, o que indica que a mesma possui proficiência em Java. Como a entidade Maria não possui esta propriedade, a mesma não pode ser considerada como proficiente em Java.

Esta regra será executada periodicamente, a cada 5 minutos. Uma vez que a entidade Maria tenha concluído todos os módulos requeridos, o perfil será atualizado como "Proficiente em Java".

Figura 27: Perfil das entidades

```
Ontology: <el>

Class: <el:Knowledge>

Class: <el:JavaProficience>
    SubClassOf:
        <el:Knowledge>

ObjectProperty: <el:hasKnowledge>

Class: <el:Entity>

Class: <el:Maria>
    SubClassOf:
        <el:Entity>

Class: <el:Andre>
    SubClassOf:
        <el:Entity>,
        <el:hasKnowledge> some <el:JavaProficience>
```

Fonte: elaborado pelo autor

4.7 Considerações sobre o Capítulo

O eProfile é um modelo para gerenciamento de perfis através de inferência em trilhas. Ao contrário da maioria das propostas de gerenciamento de perfis, o eProfile não trabalha dentro de um domínio específico, mas pode operar com qualquer tipo de informação. Ele se conecta a aplicativos que definem seus modelos de entidades e regras de inferência, e a gerenciadores de trilhas que disponibilizam a trilha das entidades e, a partir destas informações, é capaz de montar os perfis de entidade.

O Capítulo seguinte define os aspectos de implementação do protótipo e as estratégias de avaliação do modelo.

5 ASPECTOS DE IMPLEMENTAÇÃO E AVALIAÇÃO

Um protótipo foi construído para que o modelo pudesse ser avaliado. Este capítulo descreve as tecnologias utilizadas na construção do modelo (Seção 5.1) e a forma como foi realizada a avaliação (Seção 5.2).

5.1 Aspectos de Implementação

O protótipo do eProfile foi desenvolvido em Python (ROSSUM, 1995). Python é uma linguagem genérica, orientada a objetos, de alto nível, com suporte a múltiplas arquiteturas e de código aberto.

O diagrama de classes do eProfile está representado na Figura 28. As classes Comunicador, Configurador, Conversor e Raciocinador representam os respectivos agentes com os mesmos nomes, segundo descrito na Seção 4.3. Estas classes derivam da classe Agente, onde estão as tarefas comuns para todos os agentes, como inicialização, comunicação entre agentes, controle de *threads*, entre outros.

As classes nas quais estão implementados os agentes trocam mensagens com três classes de conexão. Estas classes são utilizadas para conectar com aplicações externas ou bibliotecas internas.

A classe ConectorDB conecta os agentes ao banco de dados. As regras, modelos de entidade e perfis de entidade são armazenadas no banco de dados SQLite3 (HWACI, 2013). O SQLite3 é um banco de dados SQL auto-contido transacional, disponível em domínio público. O SQLite3 foi escolhido por apresentar bom desempenho e não necessitar de um servidor separado para sua implantação.

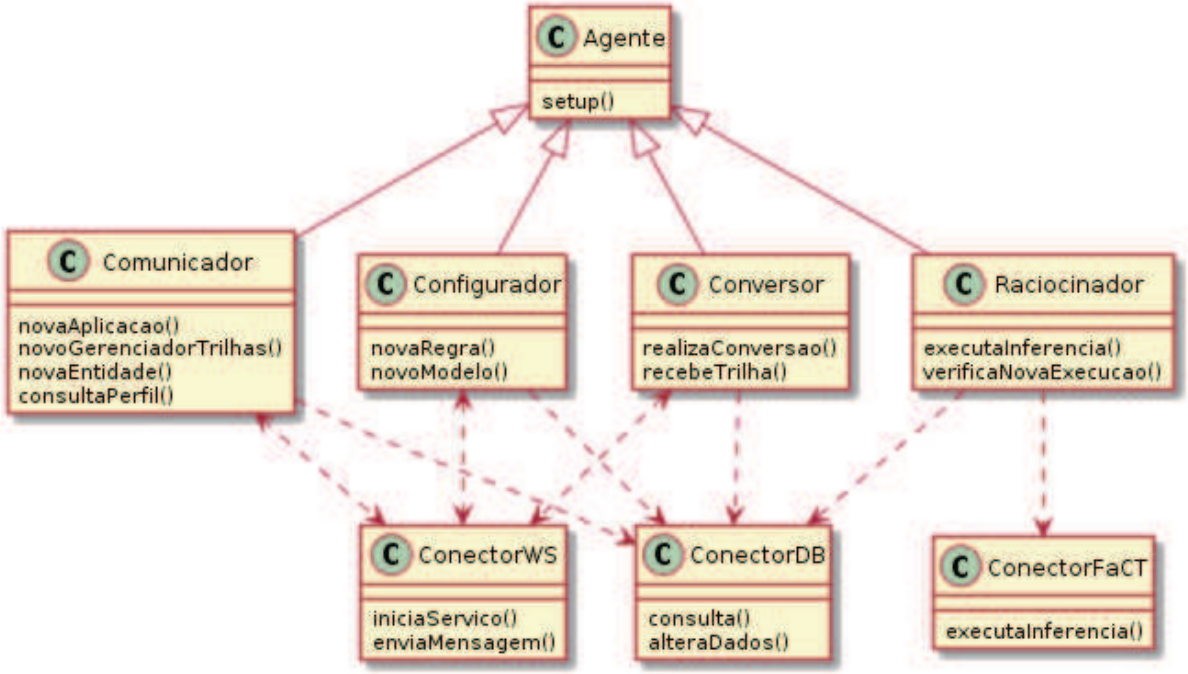
A classe ConectorWS conecta o eProfile ao gerenciador de trilhas e às aplicações, onde a comunicação ocorre através de *web services* (FIELDING, 2000). Esta classe inicializa um servidor HTTP[1] que recebe as requisições das aplicações e dos gerenciadores de trilhas. Ela também é responsável por enviar requisições HTTP às aplicações e gerenciadores de trilhas.

A classe ConectorFaCT conecta o agente Raciocinador ao motor de inferências FaCT++ (ver Seção 5.1.1). A comunicação entre o eProfile e o FaCT++ é feita através de troca de arquivos: o eProfile gera um arquivo com a ontologia OWL que será inferida, e o FaCT é executado através do sistema operacional, onde ele executa a inferência da ontologia e, como resultado, gera um arquivo com a ontologia inferida. Se houver sucesso na execução (o que pode ser verificado pela variável de retorno do sistema operacional), a classe ConectorFaCT envia o conteúdo do arquivo ao Raciocinador.

A Figura 29 mostra o diagrama de sequência do eProfile. Os textos em itálico no diagrama representam comunicação via *web services*, enquanto os textos normais representam chamadas de métodos internos do eProfile. As classes representadas neste diagrama são as mesmas da

[1] *HyperText Transfer Protocol* - Protocolo de Transferência de Hipertexto

Figura 28: Diagrama de classes do eProfile



Fonte: elaborado pelo autor

Figura 28. Neste diagrama são exibidas as cinco principais operações do eProfile: (i) inicialização, (ii) definições, (iii) recebimento de trilhas, (iv) inferência e (v) busca de perfil.

A inicialização ocorre quando uma nova aplicação ou um novo gerenciador de trilhas se conecta ao eProfile. Neste caso, a aplicação precisa se registrar para que o eProfile passe a se comunicar com ela. Os novos aplicativos e gerenciadores de trilhas são registrados na base de dados de aplicações.

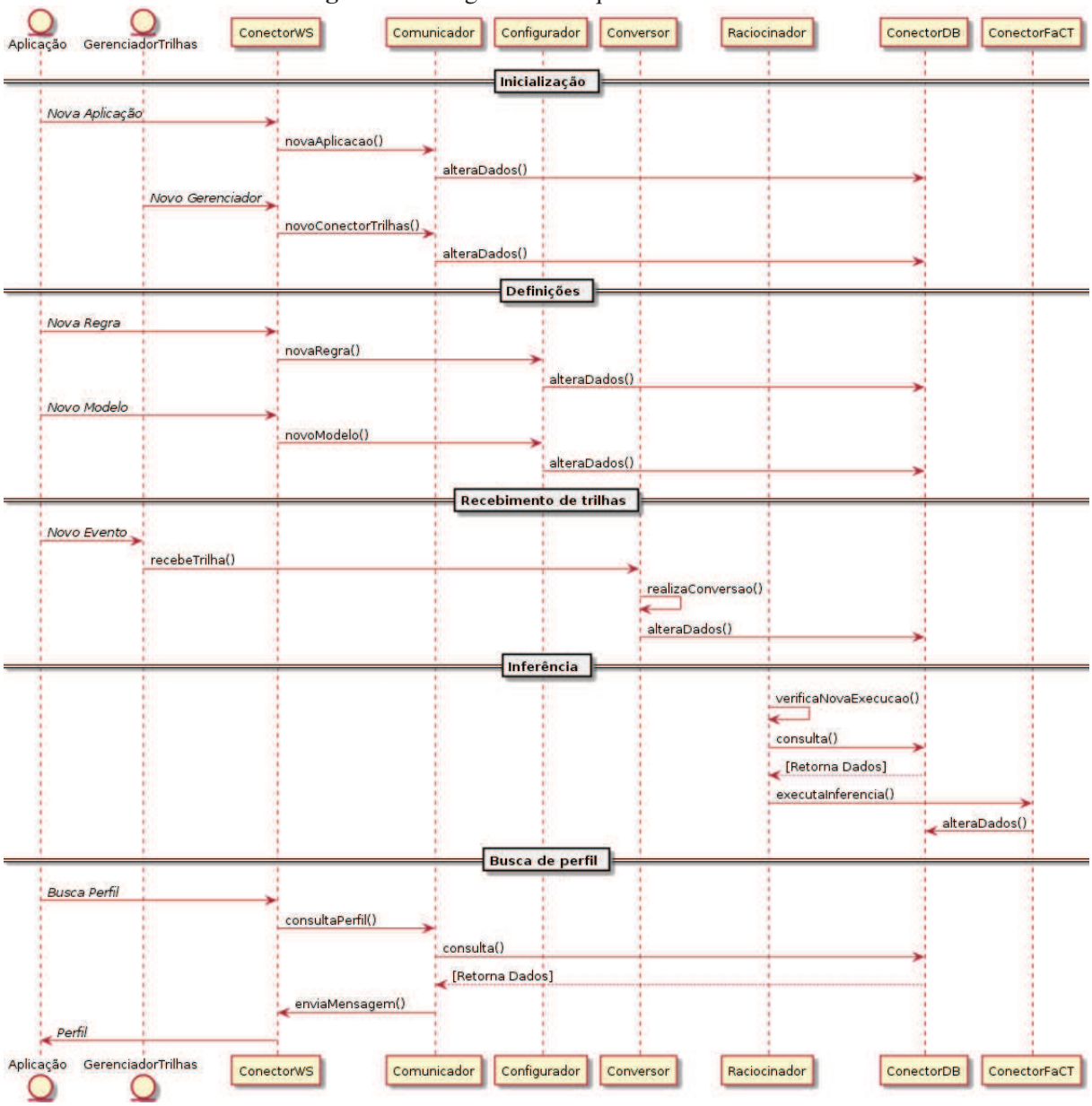
As definições são os eventos na qual uma aplicação define uma nova regra ou modelo de entidade. Estas definições podem ocorrer a qualquer momento, mesmo que a aplicação já tenha definido outras regras anteriormente. A definição pode não só criar novas regras e modelos, mas também alterar os que já foram cadastrados, ou excluí-los. Novas regras e modelos são registrados na base de dados de regras e modelos, respectivamente.

O recebimento de trilhas ocorre periodicamente, de acordo com a freqüencia configurada no eProfile. Neste caso, o agente Conversor solicita as trilhas ao gerenciador de trilhas, converte-as ao formato do eProfile e armazena-as na base de dados de trilhas.

O processo de inferência é disparado pelo agente Raciocinador. Quando o agente verifica que é o momento de executar novamente a inferência (de acordo com a periodicidade que foi definida na regra), ele chama o motor de inferência FaCT++ através da classe ConversorFaCT. Os dados são processados de acordo com as definições da regra e armazenados no banco de dados de perfil.

Periodicamente, a aplicação poderá buscar o perfil atualizado. O agente Comunicador irá

Figura 29: Diagrama de sequência do eProfile



Fonte: elaborado pelo autor

receber esta solicitação (através da classe ConectorWS), consultar o perfil na base de dados e retorná-lo à aplicação.

5.1.1 Motor de Inferência

A Figura 20 apresenta o fluxo de dados para a inferência do perfil de uma entidade. No centro desta operação está o agente raciocinador, que utiliza um motor de inferência externo para a execução das regras. Foram estudados os seguintes raciocinadores que oferecem suporte a esta operação:

- **Pellet:** é um racionador com suporte a OWL DL, que oferece bom desempenho e um *middleware* extensível. O Pellet é escrito em Java e pode ser utilizado para raciocinar com tipos de dados definidos pelo usuário e provê suporte para depuração em ontologias (SIRIN et al., 2007);

- **FaCT++:** é um raciocinador de lógica proposicional criado como uma plataforma para utilização de algoritmos de *tableaux* semântico e técnicas de otimização. Ele incorpora a maior parte das técnicas de otimização padrão (TSARKOV; HORROCKS, 2006);

- **HermiT:** é o primeiro raciocinador OWL que se baseia em um cálculo *hypertableaux*, o qual provê desempenho superior a qualquer algoritmo anterior (MOTIK; SHEARER; HORROCKS, 2009). O HermiT é disponibilizado como um plugin do Protegé, como uma ferramenta de linha de comando ou como uma biblioteca Java.

Todos os motores de inferência examinados apresentaram um conjunto de funcionalidades que supriu as necessidades do eProfile. Embora o HermiT seja apresentado como o modelo mais eficiente, nos testes realizados dentro do conjunto de funcionalidades utilizadas pelo eProfile, o FaCT++ apresentou desempenho superior, e foi selecionado como o raciocinador utilizado pelo eProfile.

5.1.2 Linguagem de Ontologia

Para o armazenamento e processamento das ontologias, foram estudadas as linguagens de representação de conhecimento CycL (LENAT; GUHA, 1989), RDF/S (W3C, 2011c), DAML+OIL (DACONTA; OBRST; SMITH, 2003) e OWL (DACONTA; OBRST; SMITH, 2003).

A linguagem **CycL** é a linguagem utilizada pelo projeto **Cyc** (LENAT; GUHA, 1989). O Cyc é um projeto de inteligência artificial que tem o objetivo de montar uma ontologia e base de conhecimento a partir do senso comum, com o propósito de permitir que aplicações de inteligência artificial realizem raciocínio semelhante ao humano. A linguagem CycL é uma linguagem declarativa baseada em lógica de primeira ordem clássica, com extensões para operadores modais e lógica de ordem superior. A base de conhecimento é dividida em micro-teorias, que

são coleções de conceitos e fatos tipicamente pertencentes a um domínio particular de conhecimentos e, ao contrário da base de conhecimento como um todo, cada micro-teoria deve estar livre de contradições.

RDF (*Resource Description Framework*) é uma linguagem utilizada para representação de metadados a respeito de recursos na Web (W3C, 2011a). No entanto, o conceito de "recurso da Web" pode ser generalizado, de forma que o RDF pode ser utilizado para representar informações a respeito de coisas que podem ser identificadas na Web, mesmo que elas não possam ser carregadas da Web. Recursos podem ser recursos eletrônicos, como arquivos, ou conceitos, como pessoas. O RDF é complementado pelo **RDF/S** (*RDF/Schema*), uma extensão semântica do RDF. O RDF/S é uma linguagem de descrição de vocabulário que provê mecanismos para descrever grupos de recursos relacionados e os relacionamentos entre estes recursos (W3C, 2011c).

O **DAML+OIL** é uma linguagem de representação de conhecimento baseada nas linguagens DAML e OIL. O DAML (*DARPA Agent Markup Language*) foi desenvolvido pelo Centro de Pesquisas Avançadas da Defesa Americana e é uma linguagem de marcação de agentes baseada no RDF. O OIL (*Ontology Inference Layer*), por sua vez, é uma linguagem com os mesmos objetivos, criada por um grupo de universidades europeias. Ambos os esforços foram unidos na criação do DAML+OIL, que por sua vez deu origem ao OWL (DACONTA; OBRST; SMITH, 2003).

O **OWL** é a linguagem de ontologia mais expressiva definida ou em definição para a Web Semântica (DACONTA; OBRST; SMITH, 2003). O OWL utiliza uma URI (*Uniform Resource Identifier*) para os nomes, e a estrutura de descrição disponibilizada pelo RDF. O OWL está uma camada acima do RDF e do RDF/S, e adiciona mais vocabulário para descrever propriedades e classes, tais como relações entre classes, cardinalidade, igualdade, tipagem mais avançada de propriedades, características de propriedades e classes enumeradas (W3C, 2011b). O OWL está disponível em três sub-linguagens: *OWL Lite*, *OWL DL* e *OWL Full*, cada uma aumentando o nível de complexidade e expressividade. Um grande número de ferramentas com capacidade para edição e raciocínio semântico com suporte para OWL está disponível no mercado.

Devido à expressividade e ao suporte de ferramentas externas, o OWL DL foi escolhido como a linguagem de ontologias utilizada pelo eProfile para modelagem das ontologia de perfis e de trilhas.

5.2 Aspectos de Avaliação

Somente é possível avaliar as funcionalidades de uma proposta de infraestrutura construindo aplicações que a utilizem e então avaliá-las, de modo a obter uma avaliação indireta do modelo (EDWARDS et al., 2003). A comunidade científica vem utilizando cenários para realizar este tipo de avaliação indireta em ambientes sensíveis a contexto (como a abordagem realizada por Dey (DEY; ABOWD; SALBER, 2001)) e ubíquos (de acordo com Satyanarayanan (SATYANA-

RAYANAN, 2001)). Este trabalho segue a mesma abordagem, tendo sido criados um protótipo e um cenário para realização de experimentos que avaliem o eProfile.

Os experimentos apresentados são baseado na geração simulada de trilhas de estudantes de graduação dentro de um contexto de Aprendizagem Ubíqua. Aprendizagem Ubíqua pode ser compreendida como um conjunto de processos educacionais que ocorrem em qualquer lugar, a qualquer tempo e com qualquer dispositivo, de forma contínua, contextualizada e integrada ao cotidiano do aprendiz (BARBOSA et al., 2011).

O objetivos dos experimentos é verificar se:

- A integração do eProfile a um modelo já existente é capaz de enriquecer a geração de informações de perfil do modelo;
- A modificação de regras de inferência de perfil durante a execução pode trazer uma melhora nos dados de perfil gerados.

As seguintes seções descrevem o ambiente da simulação (Seção 5.2.1), as aplicações utilizadas (Seção 5.2.2), a integração eProfile/LOCAL (Seção 5.2.3) e a integração eProfile/UniManager (Seção 5.2.4), as trilhas, regras e perfis gerados (Seção 5.2.5) e os parâmetros de simulação e resultados (Seção 5.2.6).

5.2.1 Ambiente de Simulação

O cenário é composto por um grupo de alunos que realiza um curso de Ciência da Computação. O curso é composto por um conjunto de 40 disciplinas, dentre as quais 12 são opcionais (o aluno deve cumprir ao menos 8 dentre estas 12). Todas as disciplinas possuem outras disciplinas como pré-requisito, exceto pelas disciplinas de início de curso (a Figura 30 exemplifica um subconjunto de disciplinas com seus pré-requisitos). Cada disciplina possui um docente, um número limitado de vagas para os alunos, e um conjunto de recursos (como salas especiais ou equipamentos).

O conjunto de disciplinas utilizado para este cenário é mostrado na Tabela 4, onde as que estão marcadas com um asterisco são opcionais. Algumas requerem um laboratório (com computadores simples), laboratório de redes e laboratório de hardware. As disciplinas sem recursos obrigatórios requerem apenas uma sala de aula.

Os alunos e docentes utilizam dispositivos móveis (*laptop* ou *smartphones*), onde estão instalados aplicativos utilizados para a administração de sua vida acadêmica. Em especial, estão instalados o UniManager e o LOCAL.

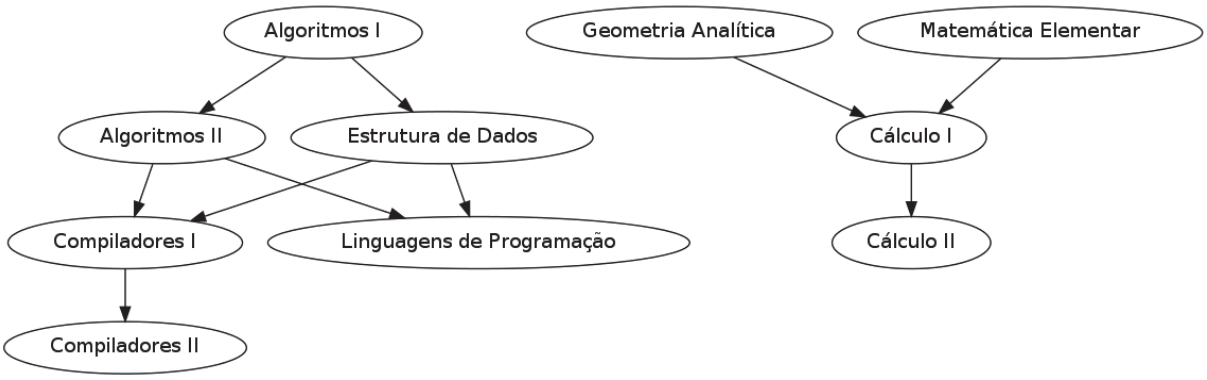
O UniManager é o sistema utilizado pela universidade para administração das disciplinas, descrito na Seção 5.2.4. Ele é usado pela administração da universidade, pelos professores e estudantes. A administração utiliza-o para cadastrar disciplinas, alunos e docentes, vincular os alunos e docentes às disciplinas e emitir relatórios gerenciais, como lista de presenças e aprovação/reprovação de alunos, bem como faltas e avaliações de professores. Os docentes

Tabela 4: Curso utilizado para o Cenário

Código	Nome	Pré-requisitos	Vagas	Recursos
A1	Algoritmos I	Nenhum	60	Laboratório
A2	Matemática elementar	Nenhum	60	–
A3	Geometria analítica	Nenhum	40	–
A4	Português	Nenhum	80	–
A5	Análise de sistemas I	Nenhum	40	–
B1	Algoritmos II	A1	40	Laboratório
B2	Estrutura de Dados	A1	30	Laboratório
B3	Cálculo I	A2, A3	40	–
B4	Análise de Sistemas II	A5	40	–
B5	Inglês	Nenhum	50	–
C1	Compiladores I	B1, B2	25	Laboratório
C2	Linguagens de Programação	B1, B2	40	Laboratório
C3	Arquitetura de Computadores	Nenhum	10	Laboratório de HW
C4	Engenharia de Software	B4	30	–
C5	Filosofia	A4	30	–
D1	Compiladores II*	C1	20	Laboratório
D2	Microprocessadores*	C3	20	Laboratório de HW
D3	Cálculo II*	B3	30	–
D4	Bancos de Dados	A5	30	Laboratório
D5	Redes*	Nenhum	20	Laboratório de redes
E1	TCC I	C1, C2, C3, C4, D4	15	–
E2	Eletrônica Digital*	D2	10	Laboratório de HW
E3	Inteligência Artificial*	C4, D3	20	–
E4	Computação Gráfica*	C4, D3	20	Laboratório
E5	Gerência de TI*	C4, D4	20	–
F1	TCC II	E1	15	–
F2	Robótica*	E2	10	Laboratório de HW
F3	Sistemas Operacionais*	D1, D2, C4	25	–
F4	Gerência de Projetos*	E5	15	–
F5	Algoritmos Avançados*	B1, B2	15	Laboratório

Fonte: elaborado pelo autor

Figura 30: Pré-requisitos de algumas disciplinas



Fonte: elaborado pelo autor

utilizam o sistema para realizar as avaliações dos alunos. Os estudantes o utilizam para realizar suas provas e trabalhos, para fazer sua matrícula nas disciplinas e para verificar suas notas e faltas. O UniManager foi criado especialmente para ser utilizado neste cenário.

Além disso, os docentes e os alunos também utilizam o LOCAL em seus dispositivos móveis. O LOCAL (BARBOSA et al., 2011) é descrito na Seção 5.2.3. O LOCAL combina as informações dos perfis com sua localização, de modo a poder assisti-los, facilitando a interação entre os aprendizes e permitindo a entrega de material pedagógico contextualizado. Os docentes utilizam o LOCAL para cadastrar o material pedagógico que será distribuído aos alunos. O LOCAL se comunica com os estudantes de modo a distribuir o material pedagógico previamente cadastrado e coloca estudantes com interesses similares em contato durante a realização de atividades pedagógicas. Caso, durante estas atividades, o estudante não tenha nenhum interesse cadastrado que corresponda àqueles requeridos pela atividade, ele tem a opção de fazer a escolha utilizando uma *interface* de usuário do LOCAL chamada Assistente Pessoal.

A Tabela 5 exemplifica uma aula de duas horas e meia envolvendo 10 alunos, na qual foi realizado um debate a respeito de Técnicas de Programação. Após o debate, os alunos realizaram um trabalho individual a respeito dos assuntos discutidos no debate. Neste exemplo, foram registrados no UniManager a presença dos alunos, a entrega do seu trabalho e a nota obtida e, no LOCAL, o conteúdo pedagógico da aula.

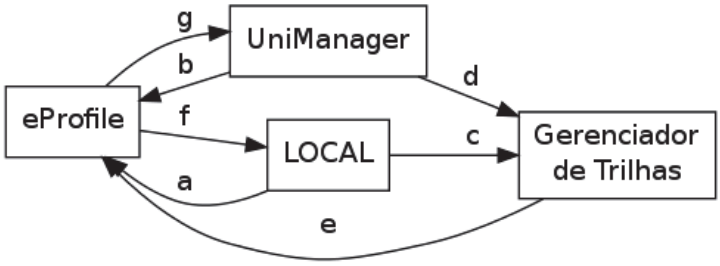
Neste exemplo, o LOCAL registrou a preferência pessoal de dois alunos, a respeito de técnicas de programação. Caso o LOCAL e o UniManager estivessem sendo usados isoladamente, esta informação teria sido apenas relevante para a tarefa imediata da aula. No entanto, a integração destes sistemas com o eProfile permite que esta informação seja utilizada na atualização do perfil dos alunos, e pode ser utilizada futuramente para facilitar a interação entre alunos e personalizar a entrega de conteúdo pedagógico (pelo LOCAL). Além disso, um perfil único pode ser criado para auxiliar o aluno na sugestão de disciplinas facultativas (pelo UniManager).

Tabela 5: Exemplo de uma aula

Evento	Horário	Ator	Ação
1	-	Professor	Insere no LOCAL o material pedagógico a respeito de banco de dados, que será utilizado na aula, e requisita a entrega de um trabalho através do UniManager.
2	7:30	Alunos	Nove alunos entram na sala.
3	7:30	LOCAL	LOCAL identifica a entrada dos alunos na sala.
4	7:30	UniManager	UniManager identifica a entrada dos alunos na sala.
5	7:30	LOCAL	LOCAL envia o conteúdo pedagógico da aula aos alunos.
6	8:00	Professor	Professor pede a criação de grupos para o debate.
7	8:00	LOCAL	O LOCAL forma três grupos baseado nas preferências dos estudantes: *Algoritmos Distribuídos* (3 alunos), *Algoritmos de Escalonamento* (3 alunos) e *Algoritmos Matemáticos* (1 aluno). Dois alunos não tem uma preferência cadastrada que permita que o LOCAL os encaixe em algum grupo.
8	8:00	LOCAL	LOCAL solicita aos dois estudantes que escolham algum grupo.
9	8:10	Alunos	Os alunos indicam sua preferência através do Assistente Pessoal do LOCAL.
10	8:10	LOCAL	O LOCAL encaixa os dois alunos nos grupos de sua preferência, um em *Algoritmos de Escalonamento* e o outro em *Algoritmos Matemáticos*.
11	8:10	Alunos	O debate inicia.
12	8:40	Alunos	Um aluno entra na aula, atrasado. O UniManager e o LOCAL registram sua entrada, mas o LOCAL não o encaixa no debate, que já está no final.
13	8:50	Alunos	O debate se encerra.
14	8:50	Professor	O professor pede que os alunos façam um trabalho em aula sobre o debate realizado.
15	10:00	Alunos	Os alunos entregam o trabalho através do UniManager e saem da sala.
16	10:00	UniManager	Recebe os trabalhos entregues e registra a saída dos alunos.
17	10:00	LOCAL	Registra a saída dos alunos.
18	-	Professor	O professor, posteriormente, avalia o trabalho dos alunos e registra a nota no UniManager.

Fonte: elaborado pelo autor

Figura 31: Modelo do Cenário



Fonte: elaborado pelo autor

5.2.2 Aplicações

O cenário é composto por quatro aplicações que interagem entre si, de acordo com a Figura 31. Ambos os softwares possuem um módulo de perfis, onde as informações de perfil são utilizadas para tomada de decisões. Como em ambos os casos o módulo de perfis é estático e depende de uma coleta de informações explícita, o eProfile é conectado a estes softwares, potencializando a criação dinâmica de perfis e a coleta implícita de informações. Além disso, o eProfile permite a interoperabilidade entre estes sistemas, permitindo que as informações de ambos sejam utilizadas na geração de um mesmo perfil. O cenário inclui também um gerenciador de trilhas, que gerencia as trilhas das entidades. Neste cenário, as disciplinas, estudantes e docentes são representados como entidades.

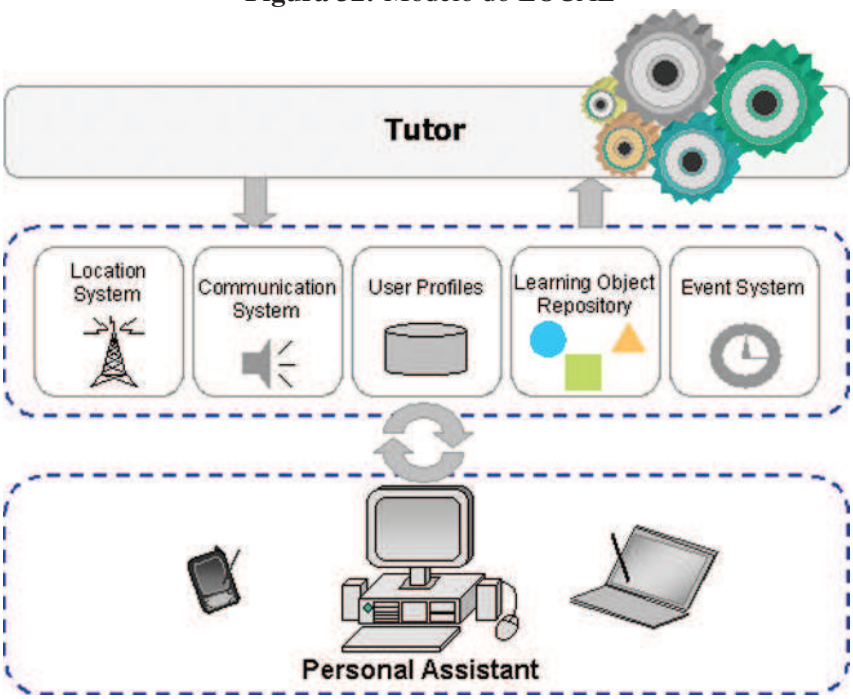
A comunicação inicial ocorre entre as aplicações e o eProfile, (setas “a” e “b”), onde as aplicações definem seus modelos de entidade e as regras. Essa comunicação pode ocorrer quantas vezes forem necessárias, pois as aplicações podem atualizar suas regras e seus modelos de entidades continuamente. Durante a execução das aplicações, as mesmas registram as regras no Gerenciador de Trilhas (setas “c” e “d”). Isto significa que as aplicações podem trabalhar desconectadas do eProfile, e conectar-se apenas no momento de atualizar alguma regra ou descarregar os dados de perfil. O eProfile busca periodicamente as trilhas no gerenciador de trilhas (seta “e”). A partir do momento que o eProfile esteja de posse das trilhas, regras e modelos de entidade, ele passa a processar estas informações, inferindo novos dados de perfil. Estes dados de perfil são então descarregados pelas aplicações (setas “f” e “g”).

5.2.3 Integração eProfile/LOCAL

O cenário integra o eProfile ao LOCAL (*Location and Context-Aware Learning*). O LOCAL usa informações de localização e de contexto como auxílio ao processo de ensino e de aprendizagem. Um sistema de localização acompanha a mobilidade dos aprendizes e, baseado nas suas posições físicas, explora oportunidades educacionais (BARBOSA et al., 2011).

O LOCAL, cuja arquitetura é mostrada na Figura 32, é formado por sete componentes: (i) um sistema de perfis de usuário, que armazena dados relevantes ao processo de ensino e

Figura 32: Modelo do LOCAL



Fonte: (BARBOSA et al., 2011)

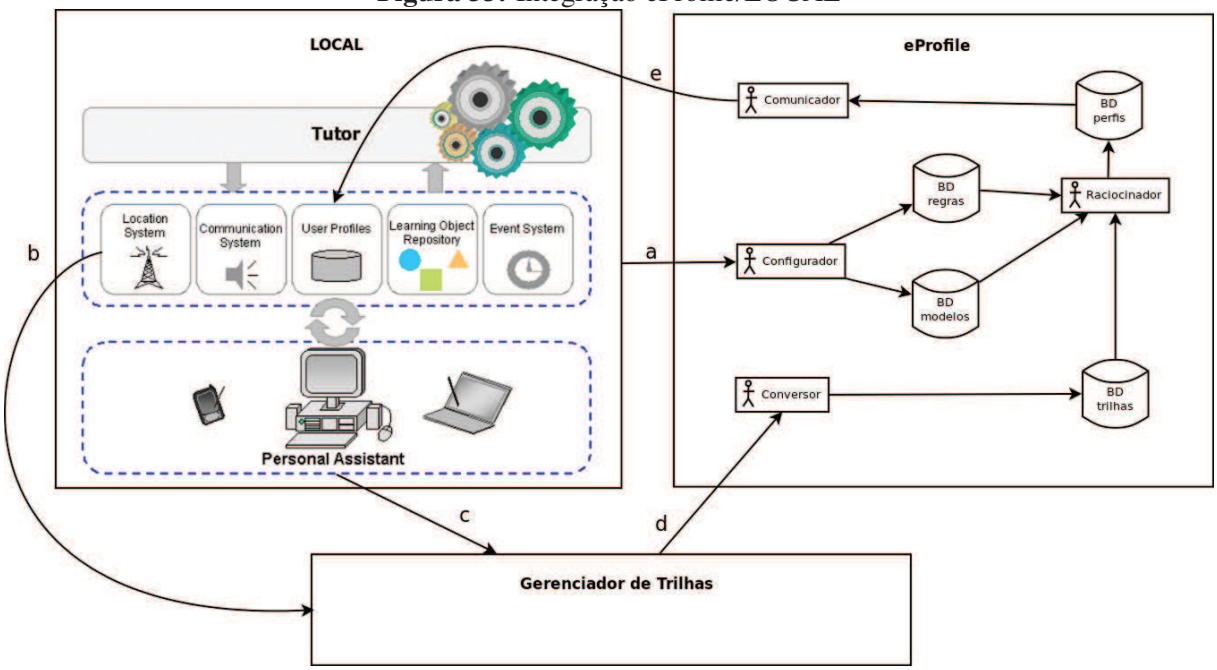
aprendizagem, (ii) um sistema de localização, (iii) um Assistente Pessoal (AP), que acompanha o usuário em seu dispositivo móvel, (iv) um repositório de objetos de aprendizagem, (v) um sistema de envio de mensagens contextualizadas, (vi) um sistema de eventos, que agenda tarefas e (vii) o Tutor, um motor de análise que realiza inferências usando dados fornecidos pelos sistemas de perfis e localização.

A coleta de informações do sistema de perfis do LOCAL é explícito. Isto significa que os dados pertinentes aos usuários (aprendizes) necessitam ser previamente cadastrados no sistema. Além disso, o sistema de perfis do LOCAL é estático, o que significa que os dados não são atualizados automaticamente em resposta às ações realizadas pelos aprendizes ou a eventos externos. Como o eProfile é um modelo de entidades dinâmico, com coleta de informações implícitas, a integração destes dois modelos resulta na potencialização das capacidades do LOCAL.

LOCAL e eProfile se integram de acordo com a Figura 33. Além do LOCAL e do eProfile, a Figura apresenta um aplicativo denominado Gerenciador de Trilhas. O Gerenciador de Trilhas é um programa externo que armazena as trilhas, processa-as e encaminha-as ao eProfile, segundo descrito na Seção 4.1. Para este cenário, foi criado um gerenciador de trilhas simples, que apenas recebe as trilhas, armazena-as e reencaminha-as ao eProfile quando solicitado.

A integração eProfile/LOCAL foi baseada nas seguintes alterações feitas no LOCAL: (i) foi incluída uma *interface* de conexão com o eProfile, capaz de gerar modelos de entidades de aprendizes e regras de processamento de perfis; (ii) foi incluída uma *interface* de conexão com o Gerenciador de Trilhas, de modo que os eventos de Localização e as ações realizadas

Figura 33: Integração eProfile/LOCAL



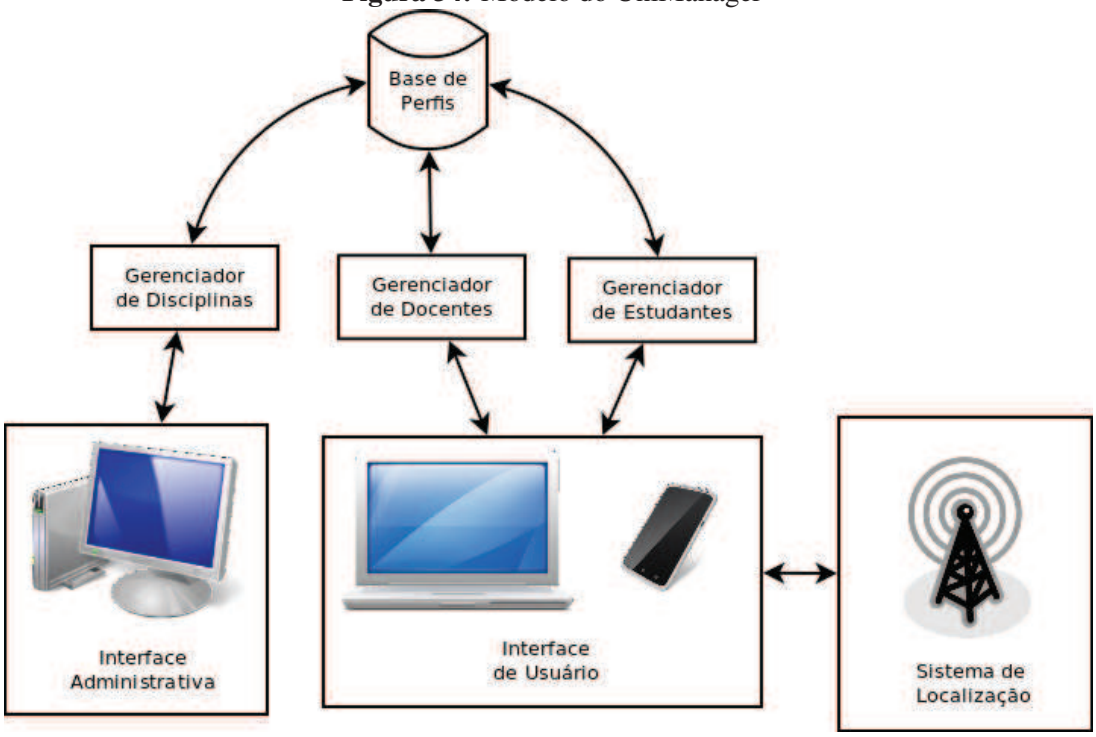
Fonte: elaborado pelo autor

no Assistente Pessoal fossem encaminhadas ao Gerenciador de Trilhas e (iii) foi incluída uma *interface* de comunicação com o eProfile, onde o módulo de Perfis de Usuário do LOCAL é alimentado pelos perfis gerados pelo eProfile.

O primeiro passo da integração se dá quando o LOCAL se comunica pela primeira vez com o eProfile. Neste momento, o LOCAL indica para o eProfile qual será o modelo de entidade esperado, e as regras para a geração deste perfil. Este conjunto de informações é registrado no eProfile através do agente Configurador (seta "a"), que organiza estas informações e registra-as nos bancos de dados de Modelos e Regras.

À medida que os aprendizes utilizam o LOCAL, as movimentações deles vão sendo identificadas pelo módulo Sistema de Localização, e as ações deles são executadas no Assistente Pessoal. Estes módulos registram os eventos no Gerenciador de Trilhas (setas "b" e "c") que, por sua vez, envia-os ao agente Conversor do eProfile (seta "d"). As trilhas enviadas são armazenadas no banco de dados de Trilhas. Estas trilhas são periodicamente processadas, de acordo com os modelos e as regras definidos inicialmente, e um processo de inferência baseado nestas informações gera os perfis, que são encaminhados ao módulo de Perfis de Usuário do LOCAL (seta "e") pelo agente Comunicador. Estes perfis servem como base para que o LOCAL conheça seus usuários (aprendizes), de forma a distribuir seu conteúdo da forma mais contextualizada possível.

Figura 34: Modelo do UniManager



Fonte: elaborado pelo autor

5.2.4 Integração eProfile/UniManager

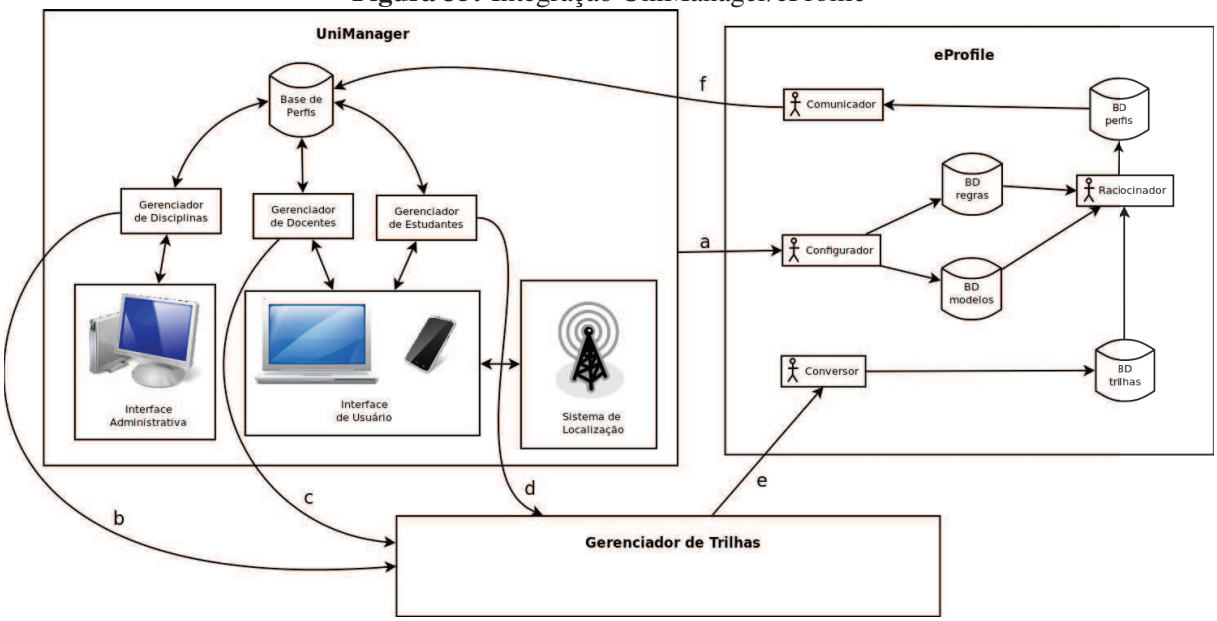
O UniManager é um *software* administrativo utilizado pela universidade para cadastramento e gerenciamento dos estudantes, docentes e disciplinas. O modelo desta aplicação é exibido na Figura 34.

O UniManager possui duas *interfaces*: administrativa e de usuário. A *interface* administrativa é utilizada pela administração da universidade para cadastrar disciplinas, docentes e estudantes. Ela é utilizada também pelos estudantes para realizar a sua matrícula para o próximo semestre. Esta *interface* conecta-se a um gerenciador de disciplinas, que gerencia as matrículas e os pré-requisitos entre as disciplinas.

A *interface* de usuário é utilizada nos dispositivos móveis dos professores e dos estudantes, e está conectada a um sistema de localização da universidade. Desta forma, todos os registros realizados no UniManager a partir desta *interface* contêm a localização do aluno dentro da universidade, isto é, a sala onde o aluno está. O sistema de localização integrado à *interface* de usuário permite verificar se o aluno ou o docente estão presentes em aula. Através desta *interface* o aluno realiza as provas e entrega os trabalhos, e o docente avalia as provas e trabalhos dos alunos.

O UniManager possui também um banco de dados de perfis. Neste banco de dados são armazenadas: (i) informações cadastrais das disciplinas, (ii) presenças e ausências dos professores e alunos, (iii) as provas e disciplinas dos alunos, e (iv) a avaliação feita pelo professor a

Figura 35: Integração UniManager/eProfile



Fonte: elaborado pelo autor

respeito dos trabalhos dos alunos.

A base de perfis é dinâmica, atualizada com base nos eventos registrados, mas está limitada aos dados registrados no próprio UniManager. A integração do UniManager ao eProfile potencializa a geração de perfis mais avançados, utilizando as informações inferidas a partir dos dados gerados pelo LOCAL. Utilizando-se destes dados, é possível registrar na base de perfis do UniManager os interesses do estudante, de modo a poder sugerir a ele disciplinas opcionais que sejam do seu interesse. Além disso, a utilização de trilhas permite que os perfis sejam gerados com base em informações históricas, obtendo informações de perfis que podem ser úteis em análises administrativas (por exemplo, qual combinação de disciplina e professor resultou historicamente na maior nota para os alunos em cada disciplina).

O UniManager se integra ao eProfile de acordo com a Figura 35. No momento da primeira comunicação entre UniManager e eProfile, o UniManager envia mensagens ao eProfile indicando qual será o modelo de entidade esperado e as regras utilizadas para a geração dos perfis (seta “a”). O registro destas informações é feito no agente configurador, responsável por validar e armazenar as regras e os modelos em seus respectivos bancos de dados. À medida que os usuários fazem uso das *interfaces* do UniManager, os dados são processados e registrados pelos módulos gerenciadores de Disciplinas, Docentes e Estudantes, que registram no Gerenciador de Trilhas os eventos ocorridos, juntamente com sua localização (setas “b”, “c” e “d”). Estas trilhas são organizadas e armazenadas no Gerenciador de Trilhas que, periodicamente, envia as trilhas ao agente Conversor do eProfile (seta “e”). O agente Conversor converte as trilhas para o formato do eProfile e armazena-as no banco de dados de trilhas. Uma vez de posse das trilhas, dos modelos e das regras, o agente raciocinador processa as informações, gerando os perfis que

são enviados ao UniManager através do agente comunicador (seta “f”). O gerenciador de trilhas mostrado na Figura 35 é o mesmo presente na Figura 33.

5.2.5 Trilhas, Regras e Perfis

Segundo descrito nas Seções 5.2.3 e 5.2.4, os aplicativos LOCAL e UniManager sofreram três alterações: (i) foram incluídas *interfaces* de comunicação com o eProfile, para que os aplicativos fornecessem os modelos de entidade e as regras, (ii) foram incluídas *interfaces* de comunicação com o Gerenciador de Trilhas, de modo que os aplicativos pudessem informar os eventos ocorridos e (iii) o módulo de perfis de ambas as aplicações foi modificado de modo a solicitar as atualizações de perfil a partir do eProfile.

O cenário é composto por três entidades: (i) estudantes, (ii) docentes e (iii) disciplinas. As entidades “estudante” e “docente” representam, respectivamente, cada um dos alunos e professores envolvidos no curso. A entidade “disciplina”, que refere-se a cada uma das disciplinas do curso, poderia ser representada tanto como uma entidade quanto como um simples recurso. Neste cenário, foi decidido que ela seria representada como uma entidade, pois desta forma é possível permitir que ela crie uma trilha, da qual pode ser obtido um perfil. Desta forma, o perfil da disciplina pode conter campos que são atualizados repetidamente, tal como a nota média da disciplina.

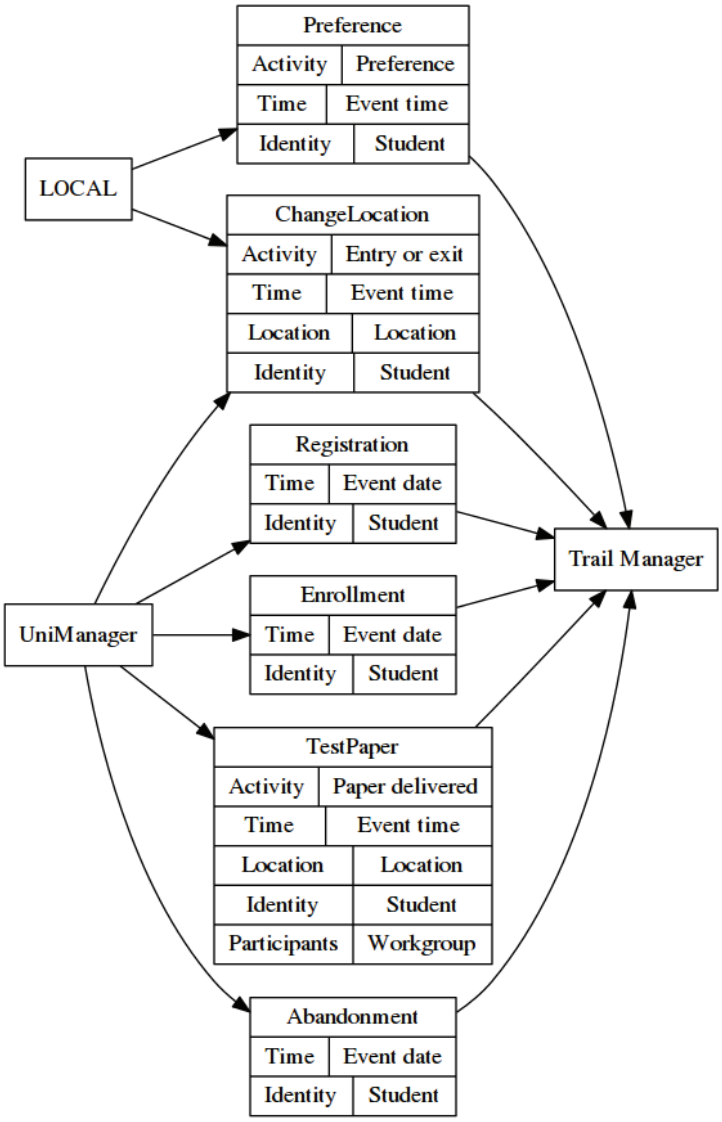
Segundo descrito na Seção 4.4.1, aplicações sensíveis a contexto estão interessadas em saber **quem**, **onde**, **quando** e **o quê** (isto é, o que a entidade está fazendo), de forma a poderem descobrir por que a situação está ocorrendo. A ontologia de trilha do eProfile, mostrada na Figura 18, define que estas questões são respondidas, respectivamente, pelas classes *Entity*, *Location*, *Time* e *Activity*. Além disso, a classe *Participants* descreve as outras entidades envolvidas neste evento. Quando uma trilha é armazenada no eProfile, a informação referente ao evento deve ser herdada destas classes, utilizando tantas classes quantas forem relevantes para que a informação esteja completa.

A trilha dos estudantes contém os seguintes eventos: cadastro, matrícula em disciplina, entrada e saída em aula, realização de provas e trabalhos, escolha de preferências e abandono de curso. O formato da trilha está detalhado na Figura 36. Na trilha dos docentes são registrados os seguintes eventos: cadastro, entrada e saída em aula e avaliação das provas e trabalhos dos alunos, segundo detalhando na Figura 37. A trilha da disciplina recebe os seguintes eventos: cadastro, início e fim de semestre e trabalhos em grupo, de acordo com a Figura 38.

O objetivo final da geração das trilhas é a geração dos perfis. Para que o eProfile tenha sucesso em gerar os perfis a partir das trilhas, ele necessita que o aplicativo informe a ele os modelos de entidades e as regras de inferência.

Os modelos de entidades são a estrutura de dados utilizada para armazenar o perfil de uma entidade. O modelo de entidade utilizado pelo módulo de Perfis de Usuários do LOCAL é mostrado na Figura 39. Os perfis dos aprendizes apresentam dois tipos de dados: (i) persistentes,

Figura 36: Ontologia de trilha do estudante



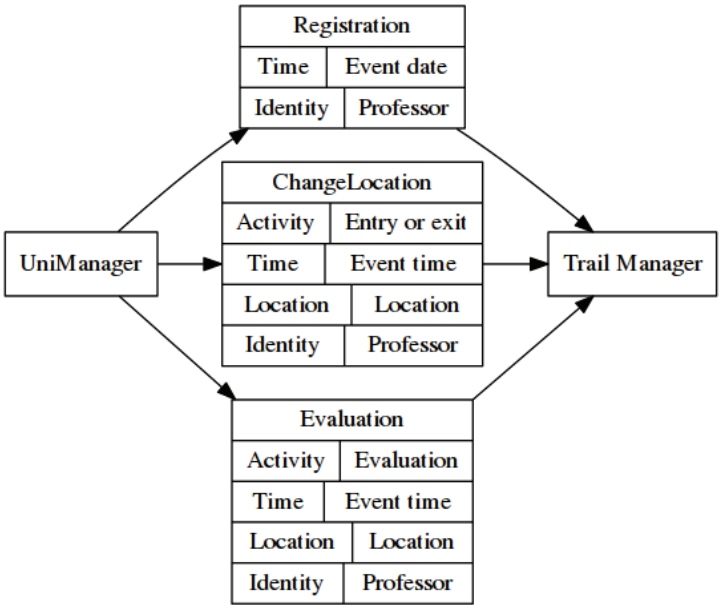
Fonte: elaborado pelo autor

relacionados ao aprendiz em todas as situações que ele se encontre e (ii) contextuais, associado com os contextos com as quais o aprendiz interage.

As seções do bloco persistente do perfil são (i) contato, que armazena dados básicos do usuário, tais como nome, telefone, entre outros, (ii) preferências, que guarda o tipo de mídia preferida pelo usuário e (iii) interesses, que identifica tópicos de interesse dos alunos. A seção de interesses está categorizada de acordo com o sistema de classificação computacional da ACM (ACM, 2012). As seções do bloco contextual são (i) relações, que identifica os relacionamentos com outros usuários dentro do mesmo contexto, (ii) desempenho, que diz respeito a objetivos que devem ser atingidos pelos estudantes e (iii) segurança, que armazena credenciais necessárias para acessar diferentes níveis em contextos específicos.

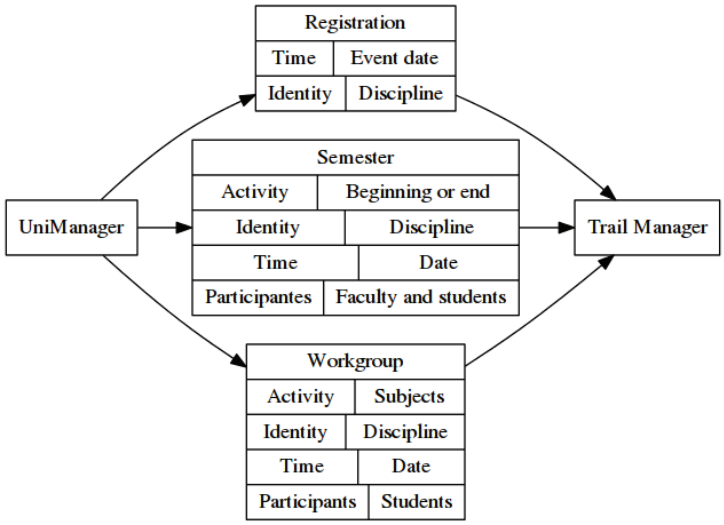
O UniManager armazena perfis de disciplinas, docentes e estudantes, e tem seu modelo de entidade ilustrado na Figura 40. As informações básicas, disciplinas matriculadas e notas/faltas

Figura 37: Ontologia de trilha do docente



Fonte: elaborado pelo autor

Figura 38: Ontologia de trilha da disciplina

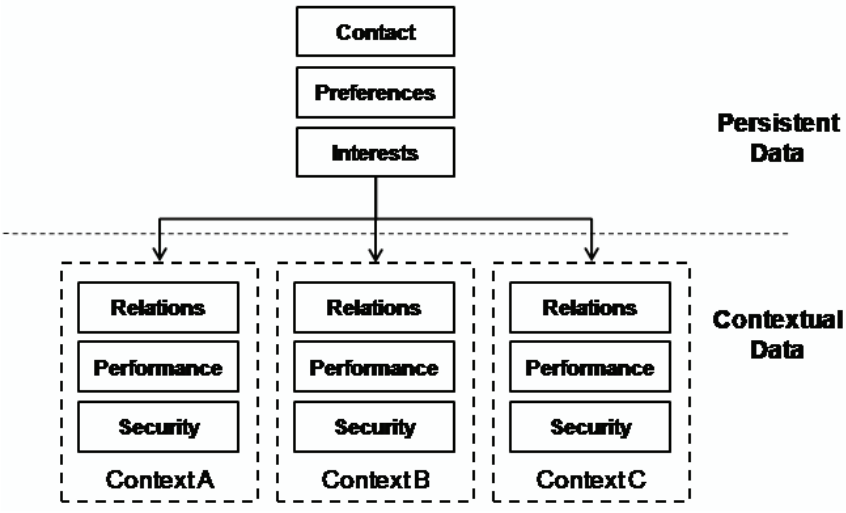


Fonte: elaborado pelo autor

do perfil do estudante são utilizadas para atividades gerenciais da universidade, como verificar se o aluno pode cursar uma determinada disciplina, assim como as informações de disciplinas do docente. As informações de desempenho do docente e de médias históricas das disciplinas são utilizadas para ajustar o docente com a disciplina que tenha maior afinidade ou desempenho. As informações de interesse dos alunos são utilizadas para sugerir disciplinas opcionais aos mesmos.

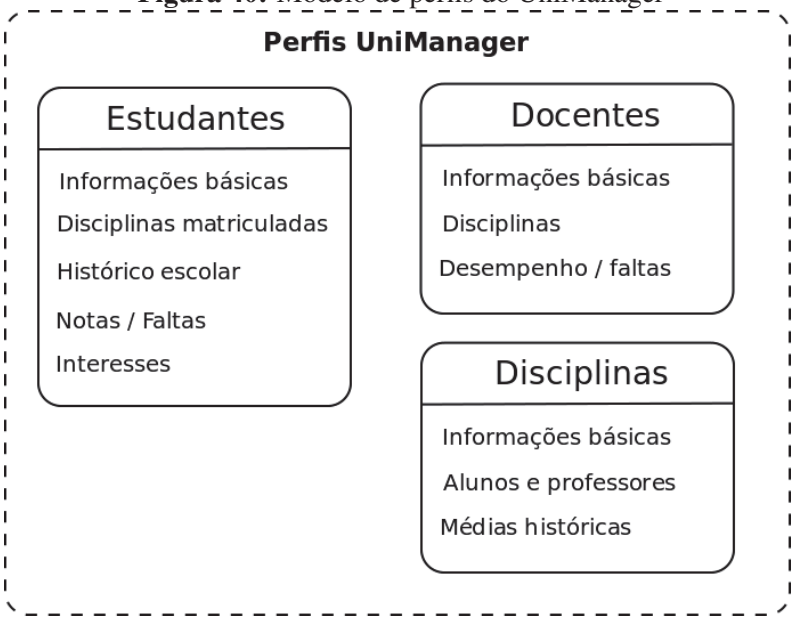
Cada aplicação define um conjunto de regras para geração dos perfis baseados nas trilhas geradas por ambos os aplicativos. Um aplicativo tem a possibilidade de utilizar a trilha de outro aplicativo para o processamento das regras e consequente geração dos perfis, permitido assim

Figura 39: Modelo de perfis do LOCAL



Fonte: (BARBOSA et al., 2011)

Figura 40: Modelo de perfis do UniManager



Fonte: elaborado pelo autor

que o perfil da entidade seja enriquecido pelo ambiente que o cerca.

A definição das regras é feita em OWL, através da criação de classes que possam ser processadas por um raciocinador. A Tabela 5 mostra o exemplo de uma aula onde o professor cadastra no LOCAL um conjunto de assuntos que serão debatidos em aula (evento 1). Este evento gera um registro uma trilha que é enviada do LOCAL ao Gerenciador de Trilhas e, subsequentemente, ao eProfile.

O LOCAL, então, tenta identificar os alunos que possuam interesses iguais aos do debate proposto, ou interesses que estejam relacionadas a estes (evento 7) através do perfil do estudante. Desta forma, o LOCAL envia as regras OWL descritas na Figura 41 para realizar esta

Figura 41: Regra OWL para identificação de interesses dos alunos

```
 1  <EquivalentClasses>
 2    <Class IRI="RelatedSubjects"/>
 3    <ObjectIntersectionOf>
 4      <Class IRI="Subject"/>
 5      <ObjectSomeValuesFrom>
 6        <ObjectProperty IRI="isRelated"/>
 7        <Class IRI="?subject"/>
 8      </ObjectSomeValuesFrom>
 9    </ObjectIntersectionOf>
10  </EquivalentClasses>
11
12  <EquivalentClasses>
13    <Class IRI="StudentsSelected"/>
14    <ObjectIntersectionOf>
15      <Class IRI="Student"/>
16      <ObjectSomeValuesFrom>
17        <ObjectProperty IRI="hasInterest"/>
18        <Class IRI="RelatedSubjects"/>
19      </ObjectSomeValuesFrom>
20    </ObjectIntersectionOf>
21  </EquivalentClasses>
```

Fonte: elaborado pelo autor

identificação. A primeira regra chama-se *RelatedSubjects* (linha 2), onde são inferidos todos os assuntos (classe *Subject*, linha 4) que sejam relacionados (linha 6) aos assuntos que serão debatidos em aula (linha 7). A expressão `?subject` é uma extensão do eProfile, que substitui esta expressão pelos assuntos que estão sendo utilizados, permitindo que a mesma regra possa ser usada em diferentes situações. Esta regra, após inferida, resulta na listagem de todos os assuntos (classes do tipo *Subject*) que se relacionam aos assuntos que serão debatidos, colocando estas classes como subclasses da classe *RelatedSubjects*.

A partir desta regra, é possível buscar os estudantes (linha 15) cujos interesses se intersectam com os interesses encontrados na classe *RelatedSubjects* (linha 18). Estes estudantes, na ontologia, passam a ser subclasses da classe *StudentsSelected* (linha 13).

O evento 7 da Tabela 5 descreve que dois alunos não se encaixaram em nenhum dos grupos pois não tinham nenhum interesse cadastrado relacionado aos assuntos do debate, e o LOCAL questionou-os em qual assunto eles gostariam de se encaixar (eventos 8 e 9), gerando assim um evento de trilha para estes dois estudantes. Uma vez que o registro tenha sido feito, o perfil dos estudantes é reprocessado de acordo com as regras definidas pelo LOCAL.

O cabeçalho da regra que atualiza o campo de interesses de um aluno é mostrado na Figura 42. Este cabeçalho define que o campo *StudentInterest* (linha 2) dos perfis dos estudantes

Figura 42: Regra para processamento do perfil do estudante

```
1  Entity:     Student
2  Field:      Interest\StudentInterest
3  Rule:       StudentInterest
4  Update:     addition
5  Frequence:  onTrailEvent(SubjectChosen)
6  Expires:    never
```

Fonte: elaborado pelo autor

Figura 43: Regra OWL para processamento de perfil de estudante

```
 1  <EquivalentClasses>
 2    <Class IRI="StudentInterest"/>
 3    <ObjectIntersectionOf>
 4      <Class IRI="SubjectChosen"/>
 5      <ObjectSomeValuesFrom>
 6        <ObjectProperty IRI="subjectChosen"/>
 7        <Class IRI="Subject"/>
 8      </ObjectSomeValuesFrom>
 9      <ObjectSomeValuesFrom>
10        <ObjectProperty IRI="hasStudent"/>
11        <Class IRI="?student"/>
12      </ObjectSomeValuesFrom>
13    </ObjectIntersectionOf>
14  </EquivalentClasses>
```

Fonte: elaborado pelo autor

(linha 1) serão atualizados com as subclasses da classe *StudentInterest*, cuja regra OWL é mostrada na Figura 43. A linha 2 define que o campo *StudentInterest* é uma classe OWL filha da classe *Interest*, de acordo com a ontologia de perfil apresentada na Figura 19. Também é definido que um novo interesse é somado ao interesses que já existem (linha 4), que esta regra é atualizada toda vez que é registrado um evento de trilha na qual um assunto é escolhido (linha 5). A regra OWL da Figura 43 define que a classe *StudentInterest* é formada dos assuntos escolhidos (linha 6) pelos estudantes que fazem parte deste contexto (linha 10 e 11) e que tenham escolhido um determinado assunto (linhas 6 e 7).

Após a aplicação da regra mostrada nas Figuras 42 e 43, a regra da Figura 41 é executada novamente, desta vez identificando os alunos que fizeram sua escolha (evento 10 da Tabela 5).

5.2.6 Protótipo e Simulação

Segundo mencionado na Seção 5.1, o protótipo do eProfile foi desenvolvido na linguagem Python (ROSSUM, 1995). Ele utiliza o FaCT++ como motor de inferência (TSARKOV; HORROCKS, 2006). A comunicação entre os três aplicativos (UniManager, LOCAL e eProfile) é realizado através de *web services* REST (FIELDING, 2000).

Para a simulação do modelo, foram desenvolvidos dois programas:

1. um simulador, que simula os estudantes e os docentes acessando o LOCAL e o UniManager;

2. um gerenciador de trilhas simulado, que armazena as trilhas geradas e disponibiliza-as para o eProfile.

Para simular o cenário de forma mais fiel possível, cada uma das entidades foi criada com um conjunto de características que as distinguem umas das outras, e faz com que tenham desempenhos diferentes e tomem decisões diferentes quando confrontadas com uma escolha. Os docentes são gerados com valores aleatórios de experiência e rigor. Estas características influem diretamente no nível de aprendizado dos estudantes, na dificuldade da disciplina e na probabilidade de desistência dos alunos. Os estudantes são gerados com níveis variados de empenho, inteligência e gostos pessoais. Estas características influenciam o desempenho do aluno, a probabilidade de desistência das disciplinas e do curso, e as escolhas que o aluno irá realizar.

O cenário simula um conjunto de 40 alunos que realiza um curso de ciência da computação em uma universidade. O tempo da simulação é aberto, sendo que a mesma é executada até que todos os alunos se formem ou tenham desistido do curso.

Uma simulação inicia com a criação das entidades de disciplinas, estudantes e docentes. São então gerados os perfis iniciais das disciplinas, e os docentes são distribuídos aleatoriamente entre as disciplinas. A partir daí, inicia-se um *loop* que termina apenas quando todos os alunos tiverem concluído ou abandonado o curso. Cada iteração do *loop* representa um semestre do curso contém os seguintes passos:

1. Os alunos escolhem as disciplinas que irão cursar e se matriculam;

2. Cada dia da disciplina é simulado, onde os alunos assistem a aula e realizam as provas ou entregam os trabalhos;

3. A nota final dos alunos é calculada, e os alunos decidem se continuarão ou abandonarão o curso.

Cada disciplina é composta por 14 aulas. Existem dois tipos de eventos que podem ocorrer em uma aula: (i) um trabalho ou debate em grupo, e (ii) um trabalho ou prova individual, sendo

que uma aula pode ter um dos dois eventos, ambos ou nenhum. A Tabela 5 exemplifica uma aula em que os dois tipos de eventos ocorreram.

Para esta simulação, foi definido que em cada disciplina foram estudados um conjunto de diferentes assuntos (3 a 5, aleatoriamente), e que cada assunto estava relacionado a 1 ou 2 outros assuntos. Por exemplo, na disciplina de "Linguagens de Programação", o assunto "Linguagem C" foi estudada, entre outras linguagens de programação. Depois, da disciplina "Estrutura de Dados", o assunto "Ponteiros" foi estudado. Como "Linguagem C" e "Ponteiros" são assuntos relacionados, os assuntos foram vinculados, e se um aluno escolhe estudar "Linguagem C" (através do LOCAL), o assunto "Ponteiros" é recomendado a ele quando ele cursar a disciplina de "Estrutura de Dados".

Disciplinas opcionais também são vinculadas a assuntos escolhidos pelos estudantes. Debates e trabalhos cujos assuntos foram escolhidos pelos estudantes através do LOCAL são utilizados para recomendar disciplinas opcionais aos alunos através do UniManager.

A Figura 44 mostra os resultados da simulação. Nesta simulação, um grupo de 40 alunos passa por um curso completo de graduação. O curso completo leva mais de 16 semestres pois, devido a um número limitado de vagas nas disciplinas, nem todos os estudantes cursaram todas as disciplinas disponíveis a eles todos os semestres. O simulador registra a porcentagem de debates e trabalhos entregues cujos assuntos foram sugeridos pelo sistema, baseado nas escolhas anteriores dos estudantes.

No primeiro semestre, o LOCAL foi capaz de recomendar assuntos através do perfil gerado pelo eProfile em 2% das situações em que os estudantes deveriam escolher um assunto. À medida que os estudantes fizeram mais escolhas, o sistema foi capaz de sugerir mais assuntos, até que o nível de recomendação estabilizou-se acima de 90% a partir do 13º semestre.

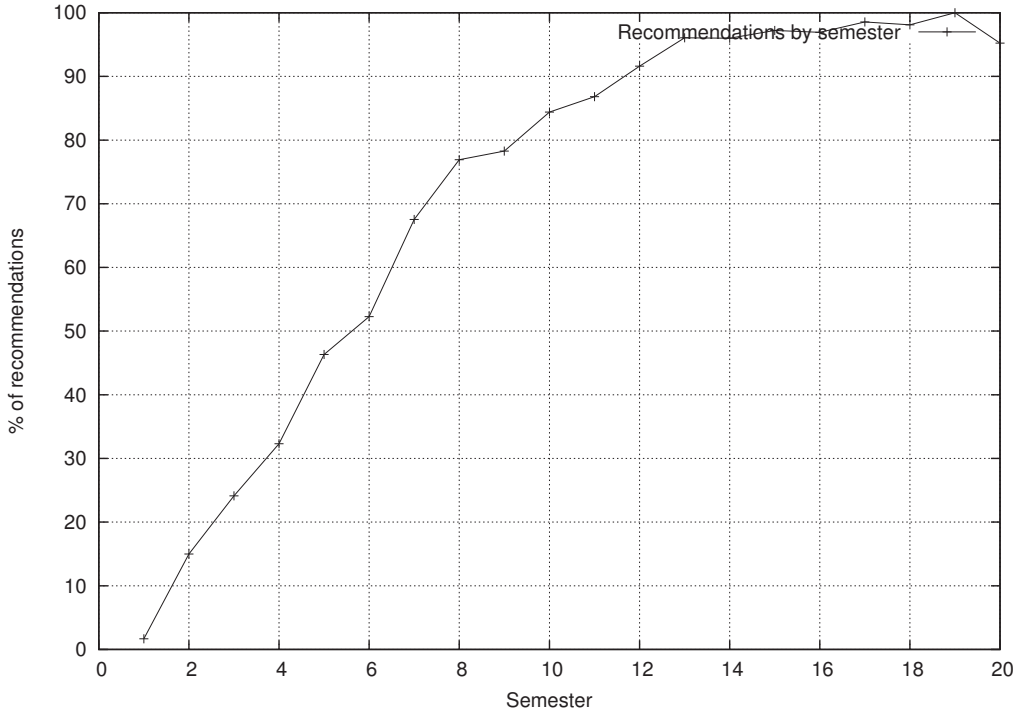
Uma das principais contribuições do eProfile é a possibilidade da alteração de regras durante a execução. Para a validação desta funcionalidade, o cenário descrito na Seção 5.2 sofreu uma modificação, na qual uma nova regra foi criada.

Esta regra baseia-se na ideia de que se um aluno fez várias escolhas iguais a outro aluno, seus interesses são semelhantes, e as escolhas feitas pelo segundo aluno podem ser utilizadas como recomendação ao primeiro. Desta forma, foi criada uma regra onde, para cada aluno, são escolhidos os 5 alunos que tomaram mais decisões iguais. Desta forma, quando o aluno é confrontado com uma escolha, as escolhas anteriores feitas por estes 5 alunos semelhantes são utilizadas como recomendação.

Para avaliar a alteração de regras durante a execução, a simulação foi executada da mesma forma e com os mesmos parâmetros que o cenário anterior, até o 7º semestre. A partir deste semestre, o LOCAL registrou esta regra no eProfile, e o eProfile passou a considerar esta regra, em adição às regras já existentes, na inferência do perfil dos alunos.

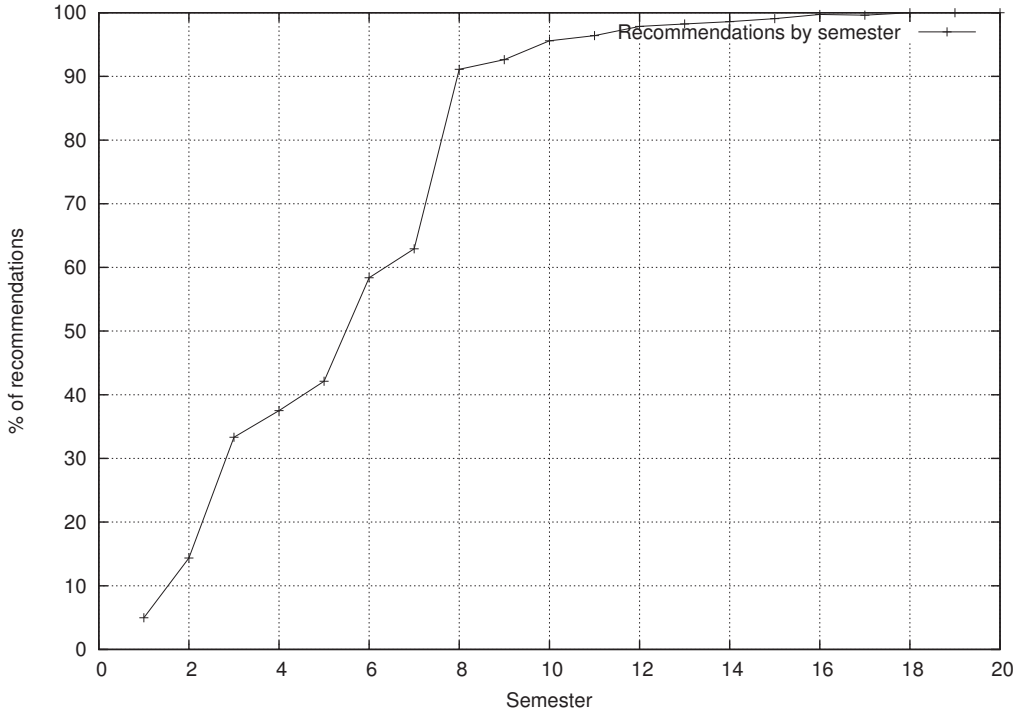
A Figura 45 mostra o resultado do segundo experimento. Comparando a Figura 44 e a Figura 45, pode-se verificar um súbito aumento na quantidade de assuntos recomendados a partir do 8º semestre, que é o resultado da modificação das regras durante a execução da simulação.

Figura 44: Resultados da simulação: recomendações automáticas por semestre



Fonte: elaborado pelo autor

Figura 45: Resultados da simulação com a alteração da regra durante a execução



Fonte: elaborado pelo autor

5.2.7 Avaliação e Contribuições

Este trabalho propôs um modelo para suportar o gerenciamento de perfis em ambientes ubíquos, permitindo a inferência de trilhas de entidades através da definição de regras pelas

aplicações. Embora o eProfile tinha sido integrado com o LOCAL e o UniManager no cenário apresentado, sua proposta é genérica e pode ser utilizada para qualquer aplicação ou conjunto de aplicações que utilize perfis para aperfeiçoamento da personalização. O uso de perfis automáticos permite que sistemas ubíquos sejam mais eficientes na compreensão da intenção do usuário, pois isso significa que sistemas deste tipo podem utilizar informações mais precisas, atualizadas e amplas para suprir as necessidades das aplicações.

O objetivo do cenário foi demonstrar que é viável integrar sistemas ubíquos com o eProfile, enriquecendo a geração de perfis destas aplicações e permitindo que elas se adaptem aos seus usuários. O cenário demonstra as quatro contribuições do modelo.

A primeira contribuição é o uso de trilhas para extração de perfis. A utilização de trilhas enriquece o banco de dados de informações que pode ser utilizado para a geração de perfis ao oferecer uma visão histórica das ações das entidades dentro dos contextos. No cenário apresentado, decisões passadas a respeito de assuntos de trabalhos e debates foram utilizadas para extrair as preferências que compuseram os perfis dos estudantes.

A segunda contribuição é a geração de perfis dinâmicos. Ao atualizar os perfis com novas informações presentes nas trilhas, é possível manter o perfil sincronizado com o que está ocorrendo com a entidade. Isto permite que as aplicações personalizem a experiência do usuário. No cenário apresentado, os perfis são atualizados toda vez que o estudante toma uma decisão referente a um debate ou trabalho.

A terceira contribuição é a capacidade de gerenciar regras de inferência dinâmicas e modelos dinâmicos. Ao permitir que as regras e os modelos sejam atualizados a qualquer momento, o eProfile facilita a adaptação da aplicação ao seu usuário. Um segundo experimento foi realizado, onde as regras de inferência para a geração dos perfis foram alteradas durante a execução da simulação, resultando numa mudança positiva no resultado do cenário.

A quarta contribuição é a operabilidade semântica do modelo. Isto abre a possibilidade de uma aplicação utilizar informações geradas por uma aplicação diferente para gerar os perfis de suas entidades. Isto é demonstrado no cenário onde o UniManager utiliza as informações geradas pelo LOCAL (as disciplinas opcionais escolhidas pelos estudantes) como parte dos dados usados para criar os perfis utilizados pelo UniManager. Esta informação depois é utilizada para sugerir disciplinas opcionais aos estudantes.

6 CONSIDERAÇÕES FINAIS

Este capítulo apresenta uma reflexão sobre as principais conclusões e contribuições do eProfile, bem como uma comparação com os trabalhos relacionados descritos no Capítulo 3. Ao final, são discutidos trabalhos futuros.

6.1 Conclusões

Este trabalho apresentou o eProfile, um modelo para gerenciamento de perfis através de inferência em trilhas. Através do estudo dos conceitos relevantes, no Capítulo 2, e de trabalhos relacionados ao modelo proposto, no Capítulo 3, foram verificados um conjunto de aspectos desejáveis a um gerenciador de perfis. Baseado nestes aspectos, foi desenvolvido um modelo, descrito no Capítulo 4, capaz de conectar-se a aplicações e gerenciar os perfis destas aplicações, inferindo novas informações de perfis através das trilhas geradas por elas.

O modelo eProfile é orientado a agentes, e possui quatro agentes que executam as operações internas (Comunicador, Configurador, Conversor e Raciocinador). Estes agentes trabalham com duas ontologias: a ontologia de trilha, que é o formato interno das trilhas dentro do eProfile, e a ontologia de perfil, que refere-se ao perfil gerado pelo agente Raciocinador. Para que um perfil seja gerado, são necessárias três informações: o modelo de entidade, as regras e as trilhas. A inferência é realizada através de um motor de inferência externo.

Foi desenvolvido um protótipo para avaliar o modelo, o que foi feito através de uma simulação (Capítulo 5) de um ambiente de aprendizagem ubíqua, onde dois softwares (LOCAL e UniManager) foram modificados para integrarem-se ao eProfile. As trilhas geradas por ambos os aplicativos foram utilizadas pelo eProfile para gerar perfis e realimentar estas aplicações com os mesmos.

Foi criado um cenário onde foram simuladas as ações de alunos dentro de um contexto de aprendizagem ubíqua. As decisões tomadas pelos alunos referente a quais assuntos estudar foram utilizadas para geração dos perfis. Os aplicativos utilizaram estes perfis para recomendar outros assuntos relacionados aos alunos. Foi verificado que a integração do eProfile a estes dois aplicativos fez com que eles pudessem recomendar um número cada vez maior de assuntos, de acordo com a passagem do tempo.

A funcionalidade de alteração de regras durante a execução foi também validada, através da aplicação de novas regras a partir do 8º semestre, o que resultou num aumento significativo de recomendações a partir deste semestre.

6.2 Contribuições

Atualmente, as áreas de computação ubíqua, sensibilidade a contexto, trilhas e personalização são bastante promissoras, sendo que um grande número de trabalhos são publicados

anualmente a este respeito. No entanto, não foi encontrado nenhum trabalho que faça uso de inferência em trilhas com o objetivo de personalizar a experiência do usuário dentro de um domínio genérico. Assim, o modelo apresentado neste trabalho busca suprir esta necessidade.

Kim e Lee (2008) apresentam um *framework* para gerenciamento de perfis baseado em *Web Services*. O modelo provê um conjunto de serviços para coletar informações necessárias para a criação de um perfil: configuração, criação, troca, processamento e serviço. O modelo também aplica metadados e técnicas de configuração dinâmica para o gerenciamento eficiente dos perfis.

O modelo apresentado por Chellouche, Arnaud e Négru (2010), assim como o eProfile, apresenta um perfil unificado que pode ser utilizado por diversas aplicações. No entanto, ao contrário do eProfile, o modelo assume a interoperabilidade sintática, o que significa que todas as aplicações precisam registrar os eventos utilizando o mesmo formato, o que pode fazer com que a aplicação não consiga registrar o evento com a riqueza de detalhes necessária. Além disso, assim como o PeLeP, o modelo de Chellouche, Arnaud e Négru (2010) está limitado a um domínio específico, que são os serviços multimídia.

O modelo UUCM (NIEDERéE et al., 2004), trabalha em um domínio genérico. Ele faz isso através da criação de uma ontologia básica que pode ser estendida pelas aplicações. Ao contrário das outras propostas (e do eProfile), o UUCM armazena os perfis no cliente, ao invés do servidor. Isto traz vantagens e desvantagens. Entre as vantagens pode-se citar a privacidade e acesso desconectado. No entanto, esta abordagem limita a interoperabilidade entre sistemas, uma vez que cada cliente fica com uma versão do próprio perfil, restringindo a utilização de um perfil unificado.

O PeLeP (LEVIS et al., 2008) suporta aperfeiçoamento automático do perfil de um aprendiz em ambientes de educação ubíqua, utilizando o histórico dos aprendizes para a atualização dos perfis através de inferências. O eProfile utiliza uma estratégia semelhante. No entanto, as regras de inferência são definidas pela aplicação, e não pelo modelo. Esta alteração é necessária para que a inferência passe de um domínio específico (educação) para um domínio genérico. Além disso, no eProfile, as regras são dinâmicas (uma aplicação pode definir ou remover regras dinamicamente), ao invés de estarem pré-definidas no modelo. Isto permite a evolução e refino das regras de inferência.

O modelo apresentado por Razmerita (2011) tem como objetivo vincular o KM (*Knowlegde Management*, ou Gerenciamento de Conhecimento) tradicional com o KM pessoal. Isto é feito através do modelo OntobUMf, capaz de gerenciar informações a respeito de usuários. A arquitetura possui três camadas: *front end*, serviços e dados. O modelo trabalha em um domínio genérico e possui três ontologias (usuários, domínios e *log*), a partir das quais são buscadas informações que classificam o usuário de várias formas, de acordo com o nível de atividade, tipo de atividade e nível de compartilhamento de conhecimento.

No artigo apresentado por Tao, Li e Zhong (2011), é proposto um modelo de ontologia para representação de conhecimento para coleta de informações personalizadas. O modelo constrói ontologias de usuário personalizadas extraindo informações globais do sistema LCSH

Tabela 6: Comparação entre o eProfile e os trabalhos apresentados

Trabalho	Interope-rabilidade	Modelos dinâmicos	Armaze-namento	Sensibilidade a contexto	Utiliza trilhas	Domínio
Kim e Lee (2008)	Sintática	Não	Servidor	Não	Não	Genérico
Chellouche, Arnaud e Négru (2010)	Sintática	Não	Servidor	Sim	Não	Multimídia
Niederée et al. (2004)	Sintática	Não	Cliente	Sim	Não	Genérico
Levis et al. (2008)	Não	Não	Servidor	Sim	Parcial	Educação
Razmerita (2011)	Sintática	Não	Servidor	Sim	Parcial	Genérico
Tao, Li e Zhong (2011)	Semântica	Não	Servidor	Sim	Parcial	Genérico
eProfile	Semântica	Sim	Servidor	Sim	Sim	Genérico

Fonte: elaborado pelo autor

e descobrindo informação a partir de repositórios locais de usuário. O sistema de informações locais é alimentado manualmente pelo usuário. Quando uma informação é necessária, o sistema faz o processamento das duas ontologias para gerar o dado solicitado.

A Tabela 6 exibe um comparativo entre o eProfile e os trabalhos apresentados no Capítulo 3, estendendo a Tabela 1 presente neste capítulo.

O eProfile fez utilização das ideias apresentadas nos trabalhos, como o aperfeiçoamento automático de perfis, a utilização de perfis unificados e o funcionamento em um domínio genérico. A estas ideias, o eProfile acrescenta como contribuição a utilização de trilhas para a extração de perfis, o que facilita a inferência com sensibilidade a contexto. Além disso, o eProfile permite que cada aplicação registre sua trilha no formato que considerar mais proveitoso, permitindo assim que a riqueza de detalhes registrados por uma aplicação possa se traduzir em um perfil mais preciso da entidade.

As principais contribuições do eProfile são a utilização de trilhas para extração de perfis, a geração de perfis dinâmicos, a capacidade de gerenciar regras de inferência dinâmicas e modelos de entidade dinâmicos, e a operabilidade semântica do modelo.

6.3 Trabalhos Futuros

O eProfile é uma proposta inicial, tal como a integração com o LOCAL e o UniManager. De acordo com os resultados obtidos na validação do modelo, apresentam-se algumas propostas para dar continuidade ao desenvolvimento e ampliação do eProfile.

Existe espaço para ampliar as avaliações do eProfile. As funcionalidades do eProfile foram validadas através da simulação de um cenário. Uma avaliação ideal envolveria a geração de perfis por entidades reais, ao invés da simulação, onde o modelo poderia ser posto à prova em um ambiente real, durante um tempo extenso. Embora a simulação traga as vantagens de várias repetições em um curto espaço de tempo, uma avaliação em um ambiente real traria um resultado mais preciso, e levaria em conta aspectos não previstos na simulação.

A simulação utilizou um conjunto pequeno de entidades, e não houve avaliação de desem-

penho. Uma vez que o modelo é genérico, outros ambientes poderiam utilizar um número maior de entidades e necessitar de perfis gerados em um período de tempo mais curto. Desta forma, a avaliação do desempenho da execução do eProfile sobre um grande número de trilhas em um tempo curto poderia apontar a necessidade de melhora de desempenho.

O cenário mostrou a integração do eProfile a duas aplicações (LOCAL e UniManager). Cada aplicação apresenta características próprias, e a integração do eProfile a outras aplicações poderá apresentar oportunidades para expansão das funcionalidades. Além disso, a integração com ambas aplicações ocorreu dentro de um contexto educacional. A integração com aplicações de outros contextos pode também indicar possibilidades para implementação de novas funcionalidades.

Como foi mencionado, o modelo é uma proposta inicial, e existem várias oportunidades para a sua ampliação. Entre elas estão a compatibilização com outros gerenciadores de trilhas, expansão das ontologias básicas, suporte à operação desconectada e/ou instável, descentralização do servidor, e suporte a protocolos de segurança e confidencialidade.

REFERÊNCIAS

ABOWD, G. D.; DEY, A. K.; BROWN, P. J.; DAVIES, N.; SMITH, M.; STEGGLES, P. Towards a Better Understanding of Context and Context-Awareness. In: HANDHELD AND UBIQUITOUS COMPUTING, 1., 1999, London, UK. **Proceedings...** Springer-Verlag, 1999. p. 304–307. (HUC '99).

ACM. **ACM Computing Classification System**. `http://www.acm.org/class/2012`.

AMATO, G.; STRACCIA, U. User Profile Modeling and Applications to Digital Libraries. In: ECDL, 1999. **Anais...** [S.l.: s.n.], 1999. p. 184–197.

BARBOSA, D.; GEYER, C.; BARBOSA, J. Globaledu - An Architecture to Support Learning in a Pervasive Computing Environment. In: RETTBERG, A.; BOBDA, C. (Ed.). **New Trends and Technologies in Computer-Aided Learning for Computer-Aided Design**. [S.l.]: Springer Boston, 2005. p. 1–10. (IFIP International Federation for Information Processing, v. 192).

BARBOSA, J. L. V.; HAHN, R. M.; BARBOSA, D. N. F.; COSTA ZANEL SACCOL, A. I. da. A ubiquitous learning model focused on learner interaction. **International Journal of Learning Technology 2011**, [S.l.], v. 6, n. 1, p. 62 – 83, 2011.

BENERECETTI, M.; BOUQUET, P.; GHIDINI, C. Contextual Reasoning Distilled. **Philosophical Foundations of Artificial Intelligence. A special issue of the journal of Experimental and Theoretical AI (JETAI)**, [S.l.], v. 12, n. 3, p. 279–305, 2000.

BRICKLEY, D.; MILLER, L. **FOAF Vocabulary Specification 0.97**. [S.l.: s.n.], 2010. Namespace document.

CARMAGNOLA, F.; CENA, F. User identification for cross-system personalisation. **Information Sciences**, New York, NY, USA, v. 179, p. 16–32, January 2009.

CHELLOUCHE, S. A.; ARNAUD, J.; NéGRU, D. Flexible user profile management for context-aware ubiquitous environments. In: IEEE CONFERENCE ON CONSUMER COMMUNICATIONS AND NETWORKING CONFERENCE, 7., 2010, Piscataway, NJ, USA. **Proceedings...** IEEE Press, 2010. p. 980–984. (CCNC'10).

CLARKE, S.; DRIVER, C. Context-aware trails [mobile computing]. **Computer**, [S.l.], v. 37, n. 8, p. 97 – 99, aug. 2004.

DACONTA, M. C.; OBRST, L. J.; SMITH, K. T. **The Semantic Web** : a guide to the future of xml, web services, and knowledge management. Indianapolis: Wiley, 2003. 281 p.

DEY, A. Understanding and Using Context. **Personal and Ubiquitous Computing**, [S.l.], v. 6, n. 1, p. 4–7, 2001.

DEY, A. K.; ABOWD, G. D.; SALBER, D. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. **Hum.-Comput. Interact.**, Hillsdale, NJ, USA, v. 16, n. 2, p. 97–166, Dec. 2001.

DRIVER, C.; CLARKE, S. Hermes: a software framework for mobile, context-aware trails. In: WORKSHOP ON COMPUTER SUPPORT FOR HUMAN TASKS AND ACTIVITIES AT PERVASIVE 2004, 2004. **Anais...** [S.l.: s.n.], 2004.

DRIVER, C.; CLARKE, S. An application framework for mobile, context-aware trails. **Pervasive Mob. Comput.**, Amsterdam, The Netherlands, The Netherlands, v. 4, p. 719–736, October 2008.

EDWARDS, W. K.; BELLOTTI, V.; DEY, A. K.; NEWMAN, M. W. The challenges of user-centered design and evaluation for infrastructure. In: SIGCHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, 2003, New York, NY, USA. **Proceedings...** ACM, 2003. p. 297–304. (CHI '03).

FIELDING, R. T. **Architectural styles and the design of network-based software architectures**. 2000. Tese (Doutorado em Ciência da Computação) — University of California, Irvine, 2000. AAI9980887.

FRANCO, L. K.; ROSA, J. H.; BARBOSA, J. L.; COSTA, C. A.; YAMIN, A. C. MUCS: a model for ubiquitous commerce support. **Electronic Commerce Research and Applications**, [S.l.], v. 10, n. 2, p. 237 – 246, 2011. <ce:title>Special Issue on Electronic Auctions: Strategies and Methods</ce:title>.

GENESERETH, M. R.; NILSSON, N. J. **Logical foundations of artificial intelligence**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1987.

GERACI, A. **IEEE Standard Computer Dictionary**: compilation of ieee standard computer glossaries. Piscataway, NJ, USA: IEEE Press, 1991.

GOLEMATI, M.; KATIFORI, A.; VASSILAKIS, C.; LEPOURAS, G.; HALATSIS, C. Creating an Ontology for the User Profile: method and applications. In: IN PROCEEDINGS OF THE FIRST INTERNATIONAL CONFERENCE ON RESEARCH CHALLENGES IN INFORMATION SCIENCE (RCIS), 2007. **Anais...** [S.l.: s.n.], 2007.

GRUBER, T. R. A translation approach to portable ontology specifications. **Knowl. Acquis.**, London, UK, UK, v. 5, n. 2, p. 199–220, June 1993.

HECKMANN, D. **Ubiquitous User Modeling**. 2005. Tese (Doutorado em Ciência da Computação) — Department of Computer Science, Saarland University, Germany, 2005.

HWACI. **SQLite Home Page**. `http://www.sqlite.org/`. Accessed in 03/11/2013.

ISO/EIC. **Database Language SQL**. `http://www.contrib.andrew.cmu.edu/ shadow/sql/sql1992.txt`. Accessed in 05/05/2012.

KIM, K.-S.; LEE, J.-D. Profile Management Framework Based on Web Services for Providing Personalized Services. In: INTERNATIONAL CONFERENCE ON CONVERGENCE AND HYBRID INFORMATION TECHNOLOGY, 2008., 2008, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2008. p. 501–508. (ICHIT '08).

KRUMMENACHER, R.; STRANG, T. Ontology-Based Context Modeling. In: IN WORKSHOP ON CONTEXT-AWARE PROACTIVE SYSTEMS, 2007. **Anais...** [S.l.: s.n.], 2007.

LENAT, D. B.; GUHA, R. V. **Building Large Knowledge-Based Systems; Representation and Inference in the Cyc Project**. 1st. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

LEVIS, D.; BARBOSA, J. L. V.; PINTO, S. C. C. S.; BARBOSA, D. N. F. Automatic Improvement of the Learner's Profile in Ubiquitous Systems. **Braz J Inform Educ**, [S.l.], v. 16, n. 1, p. 29–41, 2008.

LIU, L.; ÖZSU, M. T. (Ed.). **Encyclopedia of Database Systems**. [S.l.]: Springer US, 2009.

MOTIK, B.; SHEARER, R.; HORROCKS, I. Hypertableau Reasoning for Description Logics. **Journal of Artificial Intelligence Research**, [S.l.], v. 36, p. 165–228, 2009.

MULVENNA, M. D.; ANAND, S. S.; BüCHNER, A. G. Personalization on the Net using Web mining: introduction. **Commun. ACM**, New York, NY, USA, v. 43, p. 122–125, August 2000.

NIEDERéE, C.; STEWART, A.; MEHTA, B.; HEMMJE, M. A Multi-Dimensional, Unified User Model for Cross-System Personalization. In: E4PIA WORKSHOP 2004, 2004. **Anais...** [S.l.: s.n.], 2004.

PADGHAM, L.; WINIKOFF, M. **Prometheus**: a methodology for developing intelligent agents. 2002.

PADGHAM, L.; WINIKOFF, M. **Developing intelligent agent systems**: a pratical guide. [S.l.]: John Wiley and Sons, Ltd, 2004.

RAZMERITA, L. An Ontology-Based Framework for Modeling User Behavior - A Case Study in Knowledge Management. **IEEE Transactions on Systems, Man, and Cybernetics, Part A**, [S.l.], v. 41, n. 4, p. 772–783, 2011.

ROSSUM, G. **Python reference manual**. Amsterdam, The Netherlands, The Netherlands: CWI (Centre for Mathematics and Computer Science), 1995.

SATYANARAYANAN, M. Pervasive Computing: vision and challenges. **IEEE Personal Communications**, [S.l.], v. 8, p. 10–17, 2001.

SCHIAFFINO, S. N.; AMANDI, A. Intelligent User Profiling. In: **Artificial Intelligence**: an international perspective. [S.l.: s.n.], 2009. p. 193–216.

SCHILIT, B.; ADAMS, N.; WANT, R. Context-Aware Computing Applications. In: IN PROCEEDINGS OF THE WORKSHOP ON MOBILE COMPUTING SYSTEMS AND APPLICATIONS, 1994. **Anais...** IEEE Computer Society, 1994. p. 85–90.

SILVA, J.; ROSA, J.; BARBOSA, J.; BARBOSA, D.; PALAZZO, L. Content distribution in trail-aware environments. **Journal of the Brazilian Computer Society**, [S.l.], v. 16, p. 163–176, 2010.

SIRIN, E.; PARSIA, B.; GRAU, B.; KALYANPUR, A.; KATZ, Y. Pellet: a practical owl-dl reasoner. **Web Semantics: Science, Services and Agents on the World Wide Web**, Amsterdam, The Netherlands, The Netherlands, v. 5, n. 2, p. 51–53, June 2007.

STRANG, T.; LINNHOFF-POPIEN, C. A Context Modeling Survey. In: IN: WORKSHOP ON ADVANCED CONTEXT MODELLING, REASONING AND MANAGEMENT, UBICOMP 2004 - THE SIXTH INTERNATIONAL CONFERENCE ON UBIQUITOUS COMPUTING, NOTTINGHAM/ENGLAND, 2004. **Anais...** [S.l.: s.n.], 2004.

TAO, D.; LI, Y.; ZHONG, N. A Personalized Ontology Model for Web Information Gathering. **IEEE Transactions on Knowledge & Data Engineering**, United States of America, v. 23, n. 4, p. 496–511, 2011.

TSARKOV, D.; HORROCKS, I. FaCT++ Description Logic Reasoner: system description. In: INT. JOINT CONF. ON AUTOMATED REASONING (IJCAR 2006), 2006. **Proceedings...** Springer, 2006. p. 292–297. (Lecture Notes in Artificial Intelligence, v. 4130).

VIVIANI, M.; BENNANI, N.; EGYED-ZSIGMOND, E. A Survey on User Modeling in Multi-Application Environments. In: THE THIRD INTERNATIONAL CONFERENCE ON ADVANCES IN HUMAN-ORIENTED AND PERSONALIZED MECHANISMS, TECHNOLOGIES, AND SERVICES CENTRIC'10, 2010. **Anais...** IEEE, 2010. p. 111–116.

W3C. **SWRL**: a semantic web rule language combining OWL and ruleml. `http://www.w3.org/Submission/SWRL/`. Accessed in 05/05/2012.

W3C. **SPARQL Query Language for RDF**. `http://www.w3.org/TR/rdf-sparql-query/`. Accessed in 05/05/2012.

W3C. **OWL 2 Web Ontology Language Manchester Syntax**. `http://www.w3.org/TR/owl2-manchester-syntax/`. Accessed in 08/07/2012.

W3C. **RDF/XML Syntax Specification (Revised)**. `http://www.w3.org/RDF/`. Accessed in 05/05/2012.

W3C. **OWL Web Ontology Language Guide**. `http://www.w3.org/TR/owl-guide/`. Accessed in 03/11/2013.

W3C. **RDF Vocabulary Description Language 1.0**: rdf schema. `http://www.w3.org/TR/rdf-schema/`. Accessed in 05/05/2012.

WASFI, A. M. A. Collecting User Access Patterns for Building User Profiles and Collaborative Filtering. In: IN PROCEEDINGS OF THE 1999 INTERNATIONAL CONFERENCE ON INTELLIGENT USER INTERFACES, 1999. **Anais...** [S.l.: s.n.], 1999. p. 57–64.

WEISER, M. The Computer for the 21st Century. **Scientific America**, [S.l.], p. 94–104, 1991.

WEISER, M.; BROWN, J. S. Designing Calm Technology. **PowerGrid Journal**, [S.l.], July 1996.