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

CÁSSIA PEREIRA NINO

MaPS - Um Framework para Aplicações Colaborativas em Ambientes de Computação Ubíqua

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

Prof. Dr. Jorge Luís Victória Barbosa
Orientador

São Leopoldo, maio de 2009

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Nino, Cássia Pereira

MaPS - Um Framework para Aplicações Colaborativas em Ambientes de Computação Ubíqua / Cássia Pereira Nino. – São Leopoldo: Ciências Exatas e Tecnológicas da Unisinos, 2009.

75 f.: il.

Dissertação (mestrado) – Universidade do Vale do Rio dos Sinos. Ciências Exatas e Tecnológicas Programa Interdisciplinar de Pós-Graduação em Computação Aplicada, São Leopoldo, BR–RS, 2009. Orientador: Jorge Luís Victória Barbosa.

1. Aplicações Colaborativas. 2. Computacao Ubíqua. 3. Colaboração Ubíqua. 4. Frameworks. I. Barbosa, Jorge Luís Victória. II. Título.

"If you have an apple and I have

an apple and we exchange apples then

you and I still each have one apple.

But if you have an idea and I have

one idea and we exchange these ideas, then

each of us will have two ideas."

– George Bernard Shaw

AGRADECIMENTOS

Tenho muito o que agradecer.
Agradecer pelas ausências e presenças.
Pelas conversas e pelo silêncio.
Pelas discussões e pela concordância.
Pelos consentimentos e proibições.
Pelos momentos de diversão e seriedade.
Pela compreensão quando eu não pude justificar.
Pelo ânimo no momento certo.
Pela direção quando eu estava perdida.
Pelo conselho que eu não pedi.
Mas, principalmente, pelo apoio e confiança de todos.
Obrigada.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

API Application Programming Interface

CSCL Computer Supported Collaborative Learning

CSCW Computer Supported Cooperative Work

EJB Enterprise JavaBeans

IDE Integrated Development Environment

IM Instant Messaging

MVC Model View Controller

P2P Peer to peer

PAC Presentation Abstraction Controller

PDA Personal Digital Assistant

RMI Remote Method Invocation

XML Extensible Markup Language

RESUMO

Ao longo dos anos, os computadores cresceram em importância no suporte à colaboração. Anteriormente, utilizados apenas como ferramentas em processos já consolidados de colaboração, hoje eles também têm um papel de propulsor. Como objetivo fundamental, sistemas colaborativos visam atender totalmente uma colaboração, sob o ponto de vista do modelo 3C (comunicação, coordenação e cooperação), disponibilizando ferramentas e funcionalidades que auxiliam estas etapas. Contudo, novas tecnologias estão possibilitando cada vez mais que os domínios destes tipos de sistemas se expandam, atuando não somente nas três divisões clássicas da colaboração, mas também em etapas preliminares deste processo, como busca de usuários e recursos. Isto é possível quando estamos imergidos em um cenário de computação ubíqua, por exemplo. Fazendo uso das características que este ambiente provê, como mobilidade e ciência do contexto, é possível incrementar os processos de colaboração.

Etapa primordial de uma colaboração, a busca e definição de colaboradores é um processo pouco explorado em projetos existentes de ambientes e sistemas colaborativos, e o presente trabalho vem preencher esta lacuna. MaPS (Matching People to Share) é um framework que tem como escopo os passos elementares de uma colaboração: a busca por pessoas e por mídias de comunicação que possam ser utilizadas na interação entre os participantes. Assim, este trabalho oferece um modelo de dados e arquitetura de software que considera critérios de pesquisa e informações de contexto para a busca por pessoas em um ambiente ubíquo. O framework auxilia no desenvolvimento de aplicações colaborativas, incrementando e definindo estes processos de pesquisa.

A fim de avaliar o framework, um protótipo do MaPS foi desenvolvido e testado considerando a construção de duas aplicações que fazem uso do framework. Uma avaliação sob a perspectiva de desenvolvimento de software e outra sob a funcionalidade provida pelo framework foram geradas a fim de descrever os resultados obtidos com a elaboração dos protótipos e estudos de caso.

O diferencial do MaPS em relação a outros projetos que tratam do suporte à funcionalidade de seleção de pessoas para uma colaboração está em considerar informações distintas (não utilizando só dados que o usuário informa explicitamente), que melhoram a qualidade da busca, retornando potenciais colaboradores que são pertinentes, conside-

rando o contexto em que a busca é realizada. Dados como tipos de canais de comunicação disponíveis, perfis de usuários e localização dos mesmos não são consideradas em sua totalidade nos trabalhos relacionados, tratando-se de um aspecto que distingue MaPS dos demais trabalhos, caracterizando-o por sua principal contribuição.

Palavras-chave: Aplicações Colaborativas, Computacao Ubíqua, Colaboração Ubíqua, Frameworks.

MaPS - A Framework for Collaborative Applications in Ubiquitous Computing Environments.

ABSTRACT

Over the years, computers have grown in importance in supporting collaboration. Previously used solely as tools in already established processes, today, beyond that purpose, they also have a hole of fostering new and unforeseen forms of collaboration.

As a fundamental objective, collaborative systems intended to meet the whole set of demands of a collaboration, what means, under the terms of the 3C Model, to provide tools and features that address communication, coordination and cooperation between parts. Nevertheless, new technologies are allowing to further increase the workspace of these systems, bringing collaboration support beyond the three traditional divisions stated at the 3C Model and towards acting in preliminary stages, such as the search for users and resources. This is possible, for instance, as we are immersed in a scenario of ubiquitous computing. Using the characteristics that this environment provides, such as mobility and context-awareness, it is possible to improve the processes of collaboration.

Although a primary stage of a collaboration, the search and selection of appropriate collaboration peers is little explored in the design of existing collaborative environments and system, and this work fills this gap. In this regard, we propose MaPS (Matching People to Share) which is a framework that has as scope of work the elementary steps of a collaboration: the search for people and communication media that can be used in the interaction between participants. Hence, this work proposes a data model and software architecture that considers both search-criteria and context information while searching for people in a ubiquitous environment. The framework aids the development of collaborative applications, improving and defining these search processes. In order to evaluate the framework, we developed a prototype of MaPS and tested it considering the construction of two different applications that make use of the framework. Two assessments were then generated, one from the point-of-view of software development and another regarding the functionality delivered by framework at run-time, which are compilations of the results obtained from the development of the prototypes and carrying of case-studies.

A distinguishing aspect of MaPS, when compared to other projects the address the issue of selecting people to start or join a cooperation, is not being restricted to use only information explicitly provided by users. Instead, MaPS is able to consider alternative sources of information what improves the quality of the search, thus returning candi-

date collaboration-peers that have higher relevancy for the requesting user, considering the context in which the search is performed. Moreover, information such as about the available communication channels, as well as about user profiles and the location of users have not been investigated under an integrated perspective in related works, such is done in MaPS. We believe theses aspects form the main contribution of this work.

Keywords: Collaborative Applications, Ubiquitous Computing, Ubiquitous Collaboration, Framework.

1 INTRODUÇÃO

1.1 Motivação

Na década de 70 os microcomputadores surgiram com a promessa de prover um suporte às mais variadas atividades, incluindo-se àquelas destinadas a grupos e organizações, e então, quase uma década após, aparecem as aplicações de *groupware*. Este tipo de aplicação pode ser definido como "sistemas baseados em computador que suportam um grupo de pessoas envolvidas em um a tarefa, ou objetivo, comum e que provêem uma interface compartilhada para seus participantes" (Ellis et al., 1991).

Neste tipo de aplicação, a comunicação é um elemento chave, pois a interação entre as pessoas é permanente. Soma-se isto ao fato de a computação prover cada vez mais tecnologias para interação entre pessoas e tem-se que hoje, o computador desempenha um papel importante na comunicação entre indivíduos, não sendo exclusivo a esta categoria de sistemas. Contudo, como observado por Stahl (2002), o potencial do computador como ferramenta de suporte tende a ser mais proveitoso para grupos do que para indivíduos, uma vez que a comunicação entre os participantes sempre sofreu diversas restrições, mas que agora podem ser flexibilizadas com este tipo de suporte.

Academicamente, a exploração por *groupware* foi dividida em dois domínios separados: *Computer Supported Cooperative Work* (CSCW) e *Computer Supported Collaborative Learning* (CSCL). Enquanto CSCW foca-se nas técnicas de comunicação utilizadas para prover um suporte à cooperação, o CSCL (um sub-domínio de CSCW) preocupa-se em definir o que está sendo comunicado (Campos et al., 2003). Em geral, nas pesquisas realizadas nestes dois domínios, nota-se que CSCW é voltado para aplicações de negócios, e CSCL dirige-se para educação, procurando tornar a aprendizagem mais efetiva. Entretanto os dois domínios direcionam-se para um só objetivo: colaboração.

O suporte computacional para a colaboração é descrito pelo modelo 3C proposto por Fuks et al. (2007), onde são apresentados os três aspectos para o suporte: comunicação, coordenação e cooperação. Seguindo este modelo, em uma colaboração, os participantes devem dialogar (comunicação), organizar-se (coordenação) e trabalhar em conjunto em um espaço compartilhado (cooperação) (Gerosa et al., 2003).

Estar ciente da presença de outras pessoas, e encontrá-las, é o primeiro passo para a colaboração. Depois da descoberta, o usuário pode interagir com as pessoas e então formar grupos. Este processo de busca e seleção de usuários é crucial na colaboração (Zhang and Jin, 2005). Outro ponto crítico nestes passos elementares de uma colaboração é o tipo de mídia de comunicação utilizada (Fuks et al., 2007). Neste cenário, a computação ubíqua, em sua visão de mobilidade livre do usuário e invisibilidade da computação (Weiser, 1991), surge como um elemento potencializador, pois suas tecnologias associadas, como dispositivos móveis e adaptação ao contexto (Satyanarayanan, 2001), permitem incrementar a qualidade do processo de busca por pessoas. Destes aspectos, a mobilidade facilita o acesso do usuário a recursos dos quais ele necessita. Assim, utilizando-se desta tecnologia, criam-se novas oportunidades de interação e colaboração entre pessoas que, sem esta característica, não se encontrariam, por não pertencerem a um contexto social comum (Divitini et al., 2004). Já a ciência do contexto (Dey and Abowd, 2000) permite que possam ser utilizados elementos do ambiente no qual os participantes estão, com o objetivo de otimizar os processos de busca. Esta otimização pode ser obtida, por exemplo, em função de informações como localização dos usuários e dispositivos que eles portam. Com isso, o sistema pode configurar-se a estas informações dinamicamente de modo a adaptar-se ao usuário (Barbosa, 2007), exibindo em um resultado de busca, por exemplo, as pessoas mais indicadas para cada situação. Por sua vez, a invisibilidade garante a utilização de computação pelos usuários de modo, praticamente, imperceptível (Weiser, 1993). Com esta característica, o uso da computação nas atividades rotineiras torna-se natural e intuitivo.

O aspecto de quando e como cada um dos participantes toma conhecimento de outras pessoas no ambiente é fundamental para o processo de colaboração (Yang and Chen, 2008). No entanto, as propostas existentes não abordam o tema de forma satisfatória, de forma que este permanece como um tópico de pesquisa em aberto (Zhang and Jin, 2005).

1.2 Definição do Problema

Uma importante característica em sistemas de colaboração está em ter ciência de outros colaboradores (Carroll et al., 2003; Diamadis and Polyzos, 2004), entretanto o processo de seleção destas pessoas para uma possível colaboração é uma questão que não está completamente resolvida, como observado por Zhang and Jin (2005). Mesmo projetos posteriores ao estudo destes autores (Park et al., 2007; Yang and Chen, 2008; Farshchian and Divitini, 2009), ainda não atendem totalmente a este problema de pesquisa, especificamente, quando imerso em um ambiente ubíquo.

Chen and Yang (2006) citam em seu trabalho duas barreiras no processo de compartilhamento de conhecimentos: a dificuldade em encontrar informações relevantes e a dificuldade em encontrar colaboradores relevantes para interagir. Tanto a busca por infor-

mações quanto a busca por colaboradores são dois importantes suportes para sistemas de colaboração (Yang and Chen, 2008).

Em vista disso, um ambiente ubíquo pode oferecer ferramentas que auxiliem na solução destes problemas, como citado anteriormente. Neste tipo de ambiente, tem-se uma heterogeneidade de dispositivos e, portanto, de formas de interação, e como observado por Shafer et al. (2001), sistemas colaborativos devem cobrir estes aspectos. Sendo assim, sistemas colaborativos que estejam sendo executados neste tipo de cenário podem adaptar-se, de modo a tirar proveito destes elementos. Outras informações de contexto também podem ser utilizadas a fim de otimizar estes processos de pesquisa.

Contudo, há uma carência de soluções que abordem o problema da busca por colaboradores neste cenário, mas que não sejam específicas para um determinado domínio, e ainda, que possam ser maleáveis o suficiente para que possam ser integradas a aplicações colaborativas existentes.

Com esta visão, surge a seguinte questão de pesquisa:

- Como explorar as características oferecidas por um ambiente ubíquo, de forma a incrementar a colaboração entre pessoas?

1.3 Objetivos

Este trabalho tem como objetivo principal definir, implementar e avaliar um framework para o desenvolvimento de aplicações colaborativas em ambientes ubíquos. Este framework denomina-se MaPS (*Matching People to Share*).

A escolha por um framework como solução ao problema proposto, deve-se ao fato de o ser largamente utilizado como uma técnica de reuso de orientação a objetos (paradigma selecionado para seu desenvolvimento) (Johnson, 1997). Por ser considerado uma arquitetura semi-acabada (Pree, 1994) torna-se perfeito para o reuso, permitindo que as aplicação não só utilizem seus componentes, mas sua organização (Fontoura et al., 2000).

MaPS tem como escopo os primeiros passos de uma colaboração: a busca por pessoas e por mídias de comunicação que possam ser utilizadas na interação entre os participantes. Assim, este trabalho oferece um modelo de dados e arquitetura de software que considera critérios de pesquisa e informações de contexto para a busca por pessoas em um ambiente ubíquo. O framework auxilia no desenvolvimento de aplicações colaborativas, incrementando e definindo estes processos de pesquisa.

Para tanto, pode-se destacar como objetivos específicos:

- definir um framework que seja de propósito geral (de forma a ser utilizado no desenvolvimento de aplicações colaborativas);
- modelar o framework de modo que seja facilmente estendido, para considerar novos elementos de contexto ou mecanismos de colaboração;

- implementar o framework utilizando uma tecnologia que permita a reusabilidade, que seja de fácil manutenção e portável;
- avaliar a utilização do framework através de aplicações que o implementem;
- avaliar a utilização do framework sob duas perspectivas: na construção de software que utilize o framework (perspectiva de desenvolvimento de software) e na utilização de suas funções (perspectiva de funcionalidade).

Tendo em vista o propósito do framework, não são objetivos do trabalho:

- definir ferramentas de colaboração ou comunicação (o objetivo é propor um framework que incremente o aspecto de busca por pessoas nas aplicações, e não especificá-las);
- definir um ambiente de execução para as aplicações colaborativas;
- definir modelos de perfis de usuários (o framework utiliza perfis, contudo não define o seu modelo, pois pode adaptar-se ao modelo usado pela aplicação);
- definir modelos de contexto ou provedores de informações de contexto (o framework pode utilizar informações de contexto, entretanto não define um modelo para tal, pois pode adaptar-se às informações existentes).

1.4 Metodologia e Organização do Texto

Os capítulos seguintes refletem a metodologia adotada para o desenvolvimento da proposta.

O trabalho fundamenta-se em uma revisão bibliográfica dos principais conceitos referentes às duas áreas abrangidas pela proposta: colaboração suportada por computador e computação ubíqua, assim como a caracterização de trabalhos relevantes. Acrescenta-se ainda o estudo e apresentação de definições e fundamentos de frameworks, para contextualização da solução proposta. Este estudo encontra-se registrado nos capítulos 2 e 3. Neste sentido, o capítulo 2 apresenta uma visão histórica de colaboração, exibindo também conceitos utilizados no trabalho, como colaboração e *groupware*, CSCW e CSCL, computação ubíqua e frameworks. Já o capítulo 3, destaca projetos de pesquisa que citam ou tratam de ambientes, aplicações ou frameworks colaborativos, sendo apresentado também um comparativo de funcionalidades e características destes projetos.

A reflexão conduzida ao longo dos capítulos 2 e 3 traduz um problema de pesquisa referente à exploração do paradigma de computação ubíqua como plataforma de suporte à colaboração. Motivado por este problema, no capítulo 4 é apresentada uma proposta de framework, denominada MaPS (*Matching People to Share*), o qual visa estabelecer

mecanismos de suporte ao desenvolvimento de aplicações colaborativas em um cenário de computação ubíqua. Neste sentido, apóia-se em duas funcionalidades: busca por pessoas e seleção de canais de interação entre estas. No capítulo 4 ainda são apresentados os requisitos de ambiente no qual as aplicações são executadas, os requisitos do próprio framework, sua especificação, proposta de arquitetura e desenvolvimento do protótipo.

Subseqüentemente, o capítulo 5 aborda a metodologia de avaliação da proposta, relacionando os cenários de testes e métricas que foram utilizados. Também é apresentado o desenvolvimento desta metodologia e ainda os resultados obtidos com os experimentos.

Por fim, o capítulo 6 resgata as principais contribuições alcançadas, juntamento com algumas considerações sobre o seu desenvolvimento, apresentando também um comparativo da proposta com projetos já existentes. Neste capítulo é igualmente são apresentados trabalhos futuros que podem ser realizados como extensões do trabalho aqui apresentado.

1.5 Logotipo do Projeto

Como o próprio nome já sinaliza, o framework MaPS (*Matching People to Share*) indica a ação de pesquisa por pessoas para compartilhar. Sejam informações, idéias, ou conhecimentos, em um ambiente colaborativo, o compartilhamento é a chave para a colaboração.

Assim, o logotipo do MaPS (figura 1.1) exibe um mapa que, assim como o framework, pode auxiliar os seus usuários a encontrar pessoas em um mesmo contexto com o intuito de compartilhar e colaborar.

Figura 1.1: Logotipo do framework MaPS

2 FUNDAMENTOS: COLABORAÇÃO, COMPUTAÇÃO UBÍQUA E FRAMEWORKS

Este capítulo estabelece o pano de fundo sobre o qual o trabalho se desenvolve. Nesse sentido, inicia apresentando o conceito de colaboração suportada por computador e suas duas perspectivas principais de exploração, CSCW e CSCL. A seguir caracteriza a emergente área da computação ubíqua e conclui apresentando uma visão de convergência de ambos os cenários, denominada colaboração ubíqua (Izadi et al., 2002), a qual, acredita-se, deve potencializar a utilização de sistemas computacionais como ferramenta de suporte a colaboração.

2.1 Colaboração Suportada por Computador

A partir da década de 70, a tecnologia provida pela computação começou a permitir que os computadores pudessem ser pensados como ferramentas para automatizar as tarefas mais comuns realizadas no dia a dia de trabalho. Mas à medida que o uso da computação foi sendo expandido, novas oportunidades de utilização foram sendo pensadas. Havia uma necessidade de se pesquisar como as pessoas trabalham em grupo e organizações e como a tecnologia poderia auxiliar nestes processos (Grudin, 1994). Surge então a idéia de colaboração suportada por computador.

O significado do termo colaboração, especificamente, é algo controverso. Por fazer parte de um escopo tão multidisciplinar, nota-se que a definição de colaboração varia de acordo com o contexto, interesses e aplicações daqueles que a estão utilizando. De acordo com Yang, a colaboração, em sua forma mais usual, pode ser a interação entre os participantes ou a descoberta e compartilhamento de recursos (Yang, 2006).

Para Zhang et al. (2005a), o processo de colaboração pode ser dividido em três etapas distintas, como observado na figura 2.1.

O primeiro passo de uma colaboração é a ação de encontrar uma pessoa para se comunicar com ela e então, colaborar.

O processo de encontrar pessoas envolve uma pesquisa, onde o participante deseja buscar aqueles usuários que tenham disponibilidade e o conhecimento que ele necessita.

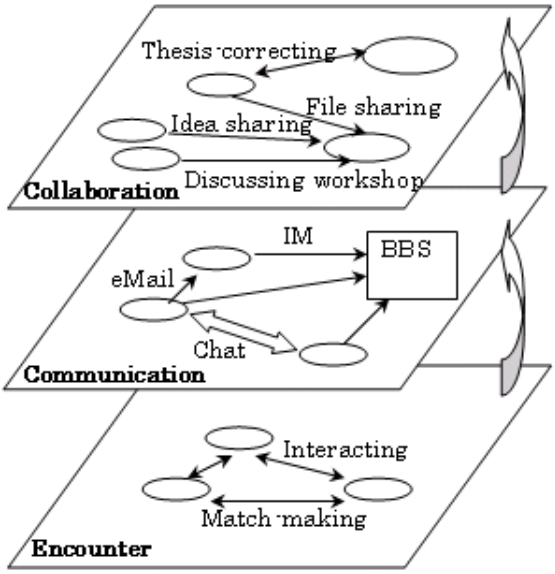


Figura 2.1: Processo de colaboração e seus passos (Zhang et al., 2005a)

A comunicação não pode ser restrita a um conjunto pré-definido de ferramentas, assim, o passo de interação deve ser suportado, independente da tecnologia empregada para tal. Por fim, tendo em vista a pessoa com quem deseja-se colaborar e o meio de comunicação empregado, dá-se a colaboração. O ato de colaborar pode ser descrito como uma troca de idéias ou arquivos, discussões ou mesmo a revisão de um documento.

Fuks et al. (2007) expandem um pouco esta idéia apresentando o modelo 3C, proposto originalmente por Ellis et al. (1991), indicando que o processo de colaboração suportado por computador é feito através de comunicação, coordenação e cooperação (figura 2.2).

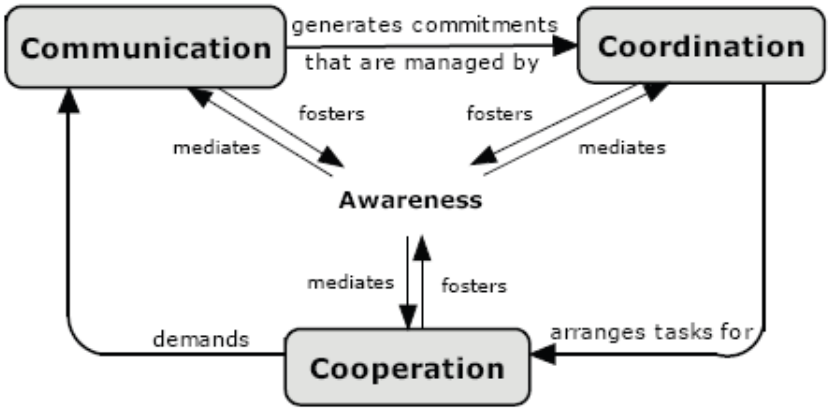


Figura 2.2: Modelo 3C (Fuks et al., 2007)

Elliott (2007) explorou definições para o termo em áreas diversas, como inteligência artificial, teoria de redes, educação e pesquisa a fim de construir uma definição para o verbete. Para este trabalho, entende-se colaboração como "o processo onde duas ou mais pessoas criam, coletivamente, representações compartilhadas de um processo e/ou

resultado que reflete a contribuição de todos os participantes" (Elliott, 2007).

Academicamente, a pesquisa por sistemas de colaboração foi separada em dois domínios diferentes: *Computer Supported Cooperative Work* (CSCW) e, posteriormente, *Computer Supported Collaborative Learning* (CSCL).

2.2 CSCW e CSCL

Na década de 80, Irene Greif e Paul Cashman organizaram um workshop, onde compareceram pesquisadores de diversas áreas, que tinha como objetivo discutir de que modo a tecnologia poderia auxiliar as pessoas a trabalharem juntas, cooperando entre si e utilizando as facilidades da computação. Foi neste evento, que o termo CSCW (*Computer Supported Cooperative Work*) foi cunhado (Bannon and Schmidt, 1989; Grudin, 1994; Penichet et al., 2007) como uma forma de se referir à pesquisa em função do trabalho em grupo suportado por computador. Como frisado por Bannon and Schmidt (1989), CSCW não é uma técnica a ser utilizada, mas sim, uma área de pesquisa. Segundo os autores ainda, o termo "trabalho cooperativo" é uma designação para um grupo de pessoas trabalhando juntas com o objetivo de produzir um produto ou serviço.

A tabela 2.1 caracteriza as diferentes modalidades de CSCW considerando as dimensões espacial e temporal do processo de colaboração.

Tabela 2.1: Matriz de classificação de sistemas CSCW (Johansen, 1988 *apud* Penichet et al., 2007)

	Mesmo Tempo	**Diferentes Tempos**
Mesmo Local	*Interações face a face* Ex.: sistemas de suporte a decisão, groupware de display único, mesa / *desktop* compartilhado, espaços interativos (*roomware*)	*Interações assíncronas* Ex.: *Team rooms*, *large group displays,* groupware para trabalho em turnos, gerenciamento de projetos
Diferentes Locais	*Interações síncronas distribuídas* Ex.: sistemas de conferência eletrônica com projeção compartilhada/única, conferência eletrônica apoiada em editores colaborativos, vídeo conferência, chats	*Interações assíncronas distribuídas* Ex.: e-mail, sistemas de mensagens estruturados, gerenciamento de workflow, controle de versão, agenda, hipertexto cooperativo (blogs, wikis)

Podemos classificar os diferentes tipos de ferramentas de auxílio à colaboração utilizando esta classificação, em função da localização dos participantes envolvidos e de suas interações.

A área de pesquisa de CSCL (*Computer Supported Collaborative Learning*) surgiu a partir de CSCW, tendo como preocupação o estudo do conteúdo que é trocado em uma colaboração (Campos et al., 2003). Neste sentido, é aplicada, comumente, na educação. Além disso, é importante observar que sistemas CSCL podem apoiar colaboração, entretanto não podem produzi-la por sua conta (Campos et al., 2003).

A função de um sistema CSCL é disponibilizar suporte para processos de aprendizagem colaborativa. O aprendizado colaborativo é definido como o trabalho realizado por um grupo com um objetivo comum (Resta, 1995).

2.3 Computação Ubíqua

Conceito introduzido por Mark Weiser (Weiser, 1991), a computação ubíqua, também referenciada como computação pervasiva (*Pervasive Computing*) surge para integrar diversas tecnologias e idéias já existentes. Diferentemente da computação baseada em *desktop*, que possui uma abordagem estática, a computação ubíqua vale-se da mobilidade do usuário através de dispositivos móveis e também de serviços providos pelas redes sem fio. De acordo com este paradigma, portanto, o usuário é livre para se mover para qualquer lugar, em qualquer tempo, utilizando o dispositivo que desejar, sendo que sempre estará inserido em um ambiente que dispõe de um poder computacional (Saha and Mukherjee, 2003; Satyanarayanan, 2001; Yamin, 2004).

E é justamente esta possibilidade de se usar a computação de forma integrada às nossas atividades do dia-a-dia que faz com que ela desapareça, do ponto de vista do usuário (Weiser, 1991). Nesse sentido, considerando a grande heterogeneidade dos dispositivos e volatilidade dos elementos que compõem o ambiente ubíquo, em grande parte causada pela mobilidade, uma das principais técnicas de suporte consiste em construir sistemas (aplicações e serviços) que são cientes do seu contexto de utilização (Schilit et al., 1994).

Um dos principais objetivos de uma aplicação ciente do contexto é atender cada um de seus usuários de uma forma personalizada. Para tanto, não apenas diferencia-os através de perfis (Syvanen et al., 2005), mas também fornece informações pertinentes a cada um destes, levando em consideração o ambiente físico em que se encontram no momento (localização) .

Um aspecto diferencial da computação ubíqua está na percepção de que os usuários, diferentemente da premissa de sistemas de computação móvel mais tradicionais, não estão atrelados ao uso de um dispositivo específico. Ao contrário, cada usuário dispõe de diversos dispositivos para interação com o ambiente de computação ubíqua, podendo escolher o que lhe for mais conveniente considerando sua situação atual. Dessa forma, in-

formações sobre um determinado contexto devem estar disponíveis não só através de um dispositivo, mas sim de todo e qualquer dispositivo que o usuário esteja usando (Divitini et al., 2004). Neste cenário, a adaptação é um elemento chave (Satyanarayanan, 1996), sendo provida em função do contexto dos usuários (Toivonen et al., 2003).

A ciência do contexto pode ser implementada combinando informações como localização do usuário e suas preferências (Naismith and Smith, 2004). Esse modelo computacional, denominado de Ciente do Contexto, se beneficia do uso de informações contextuais para aprimorar a interação com seus usuários. Uma dessas informações é a localização do usuário, fator importante para determinar a sua mobilidade. Os sistemas de localização (Hightower and Borriello, 2001) estão viabilizando o desenvolvimento de sistemas cientes da posição física do usuário (Computação Ciente da Localização). Assim, aplicações podem explorar tanto informações explícitas fornecidas pelo sistema, como também informações implícitas provenientes do contexto físico e computacional do ambiente e seus usuários. Dessa forma, essas tecnologias vêm permitindo o desenvolvimento de sistemas ubíquos cientes da localização do usuário (contexto geo-físico) e da situação onde estão inseridos (contexto social), tirando vantagem desta informação para configurar-se dinamicamente de um modo distribuído, adaptando-se as necessidades do usuário.

Estas características permitem que sejam criados cenários que propiciam a colaboração, onde o deslocamento do usuário e a adaptação do ambiente ao mesmo são fatores que auxiliam no processo de busca por colaboradores. A computação distribuída em geral e, em particular, a computação ubíqua, também permite novas oportunidades de interação entre usuários. E com isso, a colaboração entre pessoas é potencializada.

2.4 Colaboração Ubíqua

Embora as pesquisas por colaboração suportada por computador tenham iniciado quando tecnologias ubíquas não eram notórias (Farshchian and Divitini, 2009), a noção de colaboração ubíqua ganhou destaque ao longo dos últimos anos (Izadi et al., 2002). Por sua característica de heterogeneidade de dispositivos e redes sem fio, o ambiente ubíquo apresenta novas oportunidades para o trabalho colaborativo (Izadi et al., 2002).

O aparecimento da computação móvel, o amadurecimento das redes sem fio e a rápida disseminação de dispositivos de computação heterogêneos têm ajudado a formar esse modelo de colaboração, onde estes aspectos de ambientes ubíquos são suportados pelas aplicações. E a proliferação deste tipo de sistema nos mostra um crescente interesso no suporte à colaboração ubíqua (Farshchian and Divitini, 2009).

A maior motivação da colaboração ubíqua é permitir uma continuidade do trabalho colaborativo através de um acesso ubíquo à informações compartilhadas, facilidades de comunicação e serviços de *groupware*.

2.5 Frameworks

Como solução ao problema proposto (seção 1.2), optou-se por desenvolver um framework que auxilie o desenvolvimento de aplicações colaborativas em ambientes ubíquos, como já explanado. Em vista disto, esta seção se propõe a apresentar conceitos básicos e algumas metodologias existentes de desenvolvimento de frameworks, com o intuito de contextualizar o ambiente de concepção da solução apresentada.

Um framework pode ser definido como uma estrutura semi-acabada (Crespo, 2000). Esta solução incompleta pode ser preenchida através de sua instanciação, possibilitando assim, a geração de diversas aplicações dentro do domínio-alvo do framework (Fontoura, 1999).

Schmidt et al. (2001) e Pree (1994) citam também que, o fato de um framework ser projetado para ser instanciado, implica em definir uma arquitetura e oferecer construtores básicos para serem utilizados na sua instanciação. E definem, igualmente, *hot-spots*, que são pontos que devem ser extendidos de modo a caracterizar funcionalidades específicas da instância do framework.

Uma característica observada em frameworks é a sua propriedade de inversão de controle (Crespo, 2000). Ou seja, ao invés de simplesmente invocar componentes de uma biblioteca, o desenvolvedor reutiliza o programa principal (produto do framework) e decide o que será acoplado a esta estrutura (Fayad et al., 1999).

Frameworks podem ser classificados em função de diferentes aspectos, por exemplo, quanto ao domínio do problema (framework de aplicação, de domínio e de suporte); quanto à estrutura do framework (arquitetura em camadas, arquitetura de *pipes and filters*, arquitetura MVC, arquitetura PAC, arquitetura reflexiva, arquitetura microkernel, arquitetura *blackboard* e arquitetura *broker*); e quanto ao seu uso (*white-box* e *black-box*). (Crespo, 2000). Analisando-se estes aspectos, pode-se dizer que o framework MaPS é um framework de aplicação, sob o aspecto de domínio, pois tem como foco a funcionalidade de seleção de colaboradores. Sob o ponto de vista de estrutura, MaPS pode ser considerado como uma arquitetura de *pipes and filters*, em função da organização de seus componentes para o processo de busca. Quanto ao uso, o MaPS pode encaixar-se nas duas classificações (*white-box* e *black-box*), pois possui componentes que devem ser extendidos pelo desenvolvedor na instanciação do framework, e também possibilita que o desenvolvedor apenas vá agregando à sua aplicação com componentes prontos já disponíveis.

Existem diversas metodologias para o desenvolvimento de frameworks (Mattsson, 1996). Aqui, citaremos três processos de desenvolvimento de frameworks orientados a objetos.

- *Processo baseado em experiências de aplicações já desenvolvidas*. Esta metodologia parte do princípio que já foram desenvolvidas algumas aplicações no domínio do

problema (foco do framework). Assim, em função do desenvolvimento já realizado, procura-se identificar características comuns às aplicações de modo a abordá-las em um framework. A experiência ganha com a utilização do framework em aplicações posteriores ao desenvolvimento do framework serve como um aperfeiçoamento do próprio framework.

- *Processo baseado na análise do domínio*. Nesta metodologia, o primeiro passo é a análise do problema proposto, que procura identificar abstrações do domínio. Nesta etapa também são estudadas aplicações já existentes, que tem o mesmo escopo do problema (embora, isto nem sempre seja possível, pois as aplicações deste domínio podem ainda não existir). A partir das observações, o framework é construído, juntamente com uma aplicação teste a fim de avaliar o desenvolvimento do framework e refiná-lo.

- *Processo de desenvolvimento utilizando design patterns*. O primeiro passo deste processo é a construção de uma aplicação no domínio. A partir deste ponto, são identificados na aplicação *design patterns*, que serão aplicados na construção do framework.

Em função das características de um framework, podemos observar pelo menos três benefícios do uso de frameworks: modularidade, reutilização e extensibilidade (Crespo, 2000).

2.6 Considerações sobre o Capítulo

Este capítulo apresentou os conceitos primordiais que norteiam o desenvolvimento deste trabalho, colaboração suportada por computador e a visão de computação ubíqua, delineando uma visão de convergência de ambos, a colaboração ubíqua, que deve permitir ampliar as oportunidades e a qualidade dos processos colaborativos, e ainda, fundamentos de frameworks (por se tratar da solução proposta no trabalho). No próximo capítulo, esta visão preliminar é aprofundada, sendo caracterizado o estado da arte em sistemas de suporte à colaboração pela discussão de trabalhos selecionados a partir da literatura.

3 ESTADO DA ARTE EM SISTEMAS DE SUPORTE À COLABORAÇÃO

Este capítulo destaca os trabalhos que serviram como referencial bibliográfico desta proposta. As pesquisas realizadas na literatura visaram o conhecimento de serviços para ambientes colaborativos e, em especial, a seleção de pares e canais de comunicação. Ao final do capítulo, uma análise das características mais relevantes é apresentada.

3.1 uLearning

Zhang and Jin (2005) propõem um modelo de suporte à aprendizagem baseado em computação ubíqua, o uLearning. Neste modelo é definido um framework conceitual para interação social como uma facilitadora da comunicação na aprendizagem ubíqua.

O processo de interação pode ser considerado um processo linear que possui três fases: encontrar, comunicar, colaborar (Zhang et al., 2005b). Como primeira ação, o aprendiz realiza uma busca por pessoas, para após, interagir com as mesmas e com isso colaborar, contribuindo assim, para uma aprendizagem mais efetiva.

Para os autores, a aprendizagem ubíqua é composta por cinco elementos principais:

- *uEnvironment*: ambiente composto por dispositivos invisíveis e de vestir.
- *uContents*: a informação disponível no ambiente.
- *uBehaviour*: todas as ações que os aprendizes desempenham durante o processo de aprendizagem.
- *uInterface*: interface interativa entre o aprendiz e o *uEnvironment*.
- *uService*: funções de suporte, considerando teorias pedagógicas, psicológicas e sociais.

Dentre estes elementos, *uEnvironment*, *uInterface* e *uContents* estão fortemente relacionados com tecnologias de hardware (Zhang et al., 2005b), e *uBehaviour* depende da precisão dos sensores utilizados no processo. Já *uService* passa por todos outros níveis,

sendo essencial na avaliação de sistemas *uLearning*. A proposta dos autores provê este serviço, focando-se nas interações sociais dentro do ambiente.

A solução apresentada para este problema baseia-se em serviços de agentes, como pode ser visto na figura 3.1.

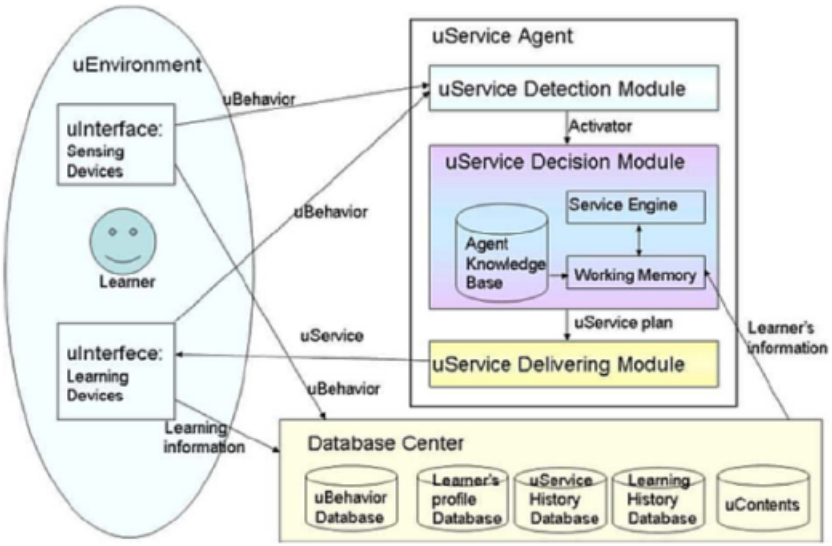


Figura 3.1: Arquitetura do *uService* (Zhang and Jin, 2005)

Os elementos de um ambiente de educação ubíqua, ou uLearning, como citam os autores estão representados na arquitetura proposta no trabalho. Esta solução é baseada em serviços de agentes com o intuito de facilitar a percepção de um aprendiz por outros que possam lhe ajudar a resolver algum problema. O agente é baseado em regras, fornecendo serviços para os aprendizes. Ele é responsável por perceber a situação do aprendiz e formar grupos entre os participantes. Toda e qualquer informação envolvida neste processo é armazenada em um *Database Center*, que por sua vez, é distribuído na rede.

O agente também auxilia o aprendiz, que tenha um problema, a encontrar pessoas que possam ajudá-lo a resolvê-lo. Ou seja, facilita o conhecimento por outros aprendizes de modo a aumentar as chances de encontrar a pessoa certa. O agente ainda indica que tipo de canal de comunicação o aprendiz pode utilizar para interagir com os colaboradores. Entretanto, na pesquisa realizada acerca deste projeto (em Zhang and Jin 2005; Zhang et al. 2005b,a) não foram encontrados mais detalhes sobre este processo de busca e indicações de pessoas e canais de comunicação para uma colaboração.

3.2 UbiCollab

UbiCollab é um ambiente de suporte para colaboração móvel (Divitini et al., 2004). Consiste de uma plataforma de serviços que provê serviços básicos para colaboração entre pessoas (Farshchian and Divitini, 2005).

Este ambiente apresenta um modelo de colaboração, que pode ser adaptado de acordo com a aplicação que se deseja desenvolver e ainda, oferece uma plataforma extensível, a fim de prover uma flexibilidade técnica. Os conceitos principais do modelo de colaboração proposto neste projeto são: usuários, atividades, recursos e ferramentas. As atividades relacionam usuários e possibilitam uma colaboração persistente; e ainda, podem conter um conjunto de recursos vinculados. As ferramentas são serviços que suportam um processo de colaboração, como por exemplo, uma ferramenta de vídeo conferência.

Neste modelo, cada entidade possui um conjunto de meta-dados, que podem ser customizados para cada aplicação, garantindo assim, flexibilidade e adaptabilidade. Estes meta-dados são divididos em dois tipos: dinâmicos e estáticos. Os meta-dados dinâmicos são atualizados com informações do ambiente em que a entidade se encontra e são específicos de cada aplicação.

Para os autores, o modelo não pode definir a relação entre as entidades, pois com isso, não estariam sendo flexíveis. Para eles, esta é uma característica que permite que desenvolvedores utilizem o modelo em diferentes domínios.

A arquitetura proposta é um modelo cliente-servidor que possui uma comunicação baseada em XML, como pode ser observado na figura 3.2. Contudo, os autores ressaltam a possibilidade de se modificar esta arquitetura para uma rede P2P, por exemplo.

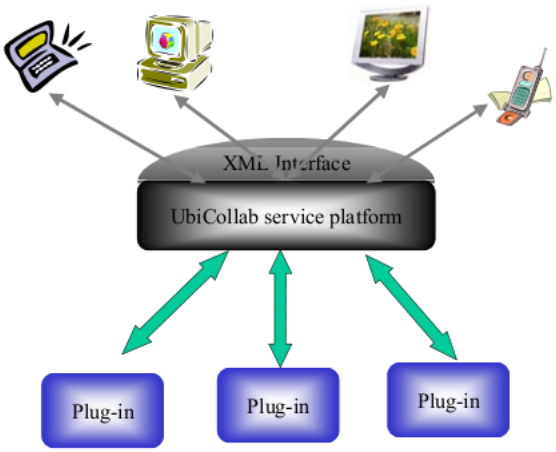


Figura 3.2: Visão geral da arquitetura do UbiCollab (Divitini et al., 2004)

A plataforma de serviços contém um conjunto de serviços que atuam como controladores e mantenedores das entidades envolvidas no ambiente (usuários, atividades, recursos e ferramentas). Além disso, é responsável pela manutenção do modelo de dados do ambiente e possui um plano de notificações em função das mudanças ocorridas nestes dados.

Além das funcionalidades já disponibilizadas para os desenvolvedores, UbiCollab tem a idéia de *plugins*, os quais são chamados de *plugins* de colaboração, que estendem as funcionalidades da plataforma de serviços da arquitetura. Estes *plugins* comunicam-se

com a plataforma através de uma interface de *web services*. A plataforma oferece também uma API de sua plataforma de serviços para seus desenvolvedores implementarem novas extensões.

Os principais serviços e *plugins* disponibilizados são *Parlay* (comunicação via *chat*), *File Server* (gerenciador de arquivos), *E-mail* (comunicação via e-mail), *Database* (persistência de informações), *Location* (localização de usuários) e *Position Service* (posição absoluta dos usuários) (Divitini et al., 2004; Farshchian and Divitini, 2005).

3.3 Peer-to-peer e redes sociais

Peer-to-peer (P2P) e redes sociais, estes elementos formam a base da proposta de Chen and Yang (2006). Os autores aplicam o conceito de redes peer-to-peer e a tecnologia de mensagens instantâneas para a distribuição de conhecimento em espaços virtuais de colaboração. O objetivo é a criação de redes sociais para incrementar a busca por recursos e formação de grupos. Contudo, antes da formação de grupo é importante a classificação de pessoas, baseando-se em diversos atributos como: relações sociais, interesses pessoais, domínio de conhecimentos e proficiência em determinado assunto.

Duas grandes barreiras que existem no compartilhamento de informações é a determinação das informações mais relevantes e também dos pares mais relevantes em uma colaboração (Yang and Chen, 2008). Neste trabalho, estes dois problemas são tratados, já que são considerados como um importante suporte para aplicações colaborativas.

A aplicação de redes sociais em uma rede P2P foi utilizada para resolver os dois problemas indicados pelos autores. Esta solução pode auxiliar pessoas a encontrar recursos (informações) relevantes e de qualidade, assim como encontrar potenciais colaboradores.

Para encontrar pessoas mais indicadas em cada situação de colaboração foi desenvolvido um mecanismo de classificação de pessoas baseado na rede social especificada. Estas redes são formadas a partir do perfil social dos participantes. Um exemplo gráfico de rede social pode ser visto na figura 3.3.

Nesta rede, cada nodo é uma pessoa e as arestas que ligam-nas possuem valores que representam uma distância social. A distância social entre duas pessoas é determinada por uma equação que relaciona um conjunto de valores como por exemplo, o número de relações entre os participantes (Chen and Yang, 2006).

A principal busca feita neste ambiente é por recursos, que são conteúdos de conhecimento ou pessoas. Como os usuários estão interligados por uma rede P2P, onde cada um dos participantes tem em seu poder um conjunto de recursos, a busca sempre se dá entre os pares. Para otimizar esta busca, de modo a retornar resultados eficientes, é utilizada a rede social. O resultado dessa busca é uma lista de pessoas, cada uma com um peso diferente, onde quanto maior o peso, mais relevante esta pessoa pode ser para uma colaboração. Nesta pesquisa, os usuários podem informar palavras chaves para restringir

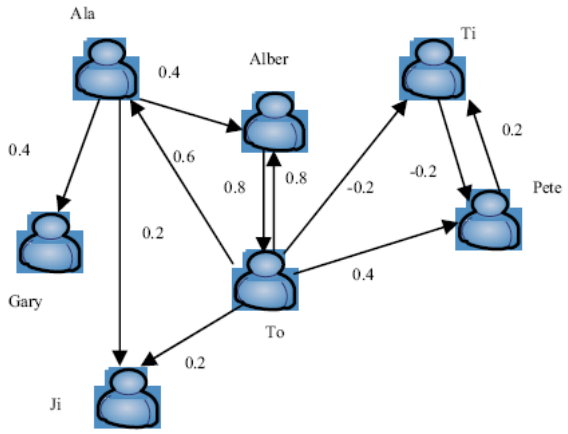


Figura 3.3: Rede social e suas distâncias sociais (Chen and Yang, 2006)

o resultado.

Para determinar a capacidade de colaboração de cada um dos participantes (informação que é levada em consideração na hora de se realizar uma busca), os autores consideraram dois aspectos (Yang and Chen, 2008): domínio de conhecimento e reputação de contribuições. O valor que indica o domínio de conhecimento é fornecido manualmente para o sistema através de usuários que testam a proficiência dos participantes. Já, o cálculo da reputação de cada um dos usuários é realizado em função de suas colaborações passadas.

O trabalho não especifica como a integração desta rede social se dá em função de outras aplicações, apenas indica o seu funcionamento em uma aplicação com fins de teste.

3.4 Collaborator

Em Bergenti et al. (2003), é apresentado o projeto Collaborator (*Collaborative Framework for Remote and Mobile Users*). Este é um projeto de pesquisa de dois anos financiado por um consórcio de entidades que tem como objetivo principal a criação de um ambiente de software que proveja um espaço virtual de compartilhamento e que suporte as atividades de equipes virtuais. Outro objetivo também é explorar os benefícios da integração do ambiente proposto (o framework Collaborator) com tecnologias de computação móvel.

A principal motivação do projeto é o fato da crescente necessidade de se acessar informações de qualquer lugar, e para tanto, o projeto é direcionado a ambientes ubíquos (Bergenti et al., 2002). Elementos como mobilidade, adaptabilidade a diferentes dispositivos e sistemas operacionais, e diversas formas de comunicação são explorados pelo projeto.

Collaborator é implementado em Java e utiliza várias tecnologias específicas para esta linguagem como JBoss, MySQL, JADE e conceitos de EJB (*Enterprise Java Beans*),

portlets, *servlets* entre outros. Na figura 3.4, pode ser observada a arquitetura proposta para o framework, relacionando seus principais elementos e tecnologias.

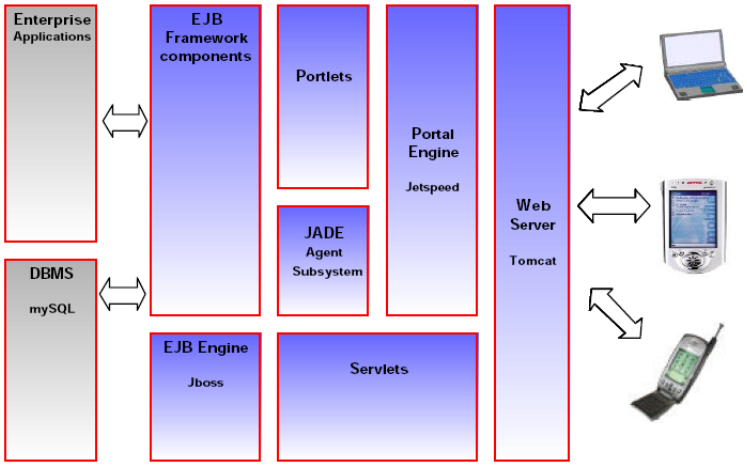


Figura 3.4: Arquitetura do framework Collaborator (Bergenti et al., 2003)

O meio de comunicação usado é a *web* por se tratar de um meio amplamente difundido e suportado por diversos tipos de dispositivos. Também são exploradas tecnologias de agentes a fim de incrementar os mecanismos de comunicação oferecidos pela *web* para suportar um compartilhamento síncrono de aplicações.

O sistema Collaborator define o conceito de *workspaces* compartilhados, que são um espaço virtual onde os participantes podem compartilhar aplicações, trocar dados e comunicar-se. Estas ações podem ser feitas pelos usuários dentro de uma sessão colaborativa, que é criada no ambiente pelos usuários ditos coordenadores. No ambiente, os usuários podem se associar a sessões a qualquer momento e então compartilhar informações com outros participantes.

Sob um aspecto funcional, Collaborator pode ser dividido em três partes distintas: framework, serviços e portal. O framework é composto por quatro sub-sistemas principais que têm como objetivo gerenciar usuários, sessões, recursos e a execução de aplicações (Bergenti et al., 2003). Os serviços oferecidos suportam atividades colaborativas como: reuniões virtuais, apresentações em grupo, calendários de grupo, agenda pessoal, *chat*, e gerenciamento de reuniões. Por fim, o portal é a aplicação que o usuário final utilizará para acessar recursos e informações.

O sistema também faz uso de perfis de usuários, de modo que apresenta, através do portal, informações personalizadas para cada usuário, com base em suas preferências e interesses. Neste perfil ainda são armazenadas informações como características da rede onde o usuário encontra-se e do dispositivo que está sendo utilizado.

Por considerar a comunicação entre os participantes como um elemento importante para colaboração, Collaborator também provê suporte para tal, sendo que esta pode ser síncrona e assíncrona. Porém, os sistema não oferece um serviço que auxilie o usuário a

identificar qual o melhor meio de comunicação a ser utilizado em uma interação.

3.5 Comparação entre Projetos

Naturalmente, os projetos estudados apresentam funcionalidades que não são idênticas. Contudo, observa-se que todos partilham a preocupação com o tema de pesquisa do presente trabalho: seleção de pessoas e canais de comunicação em processos de colaboração. Para um bom entendimento das contribuições e limitações de cada um destes projetos, é apresentada uma tabela comparativa (tabela 3.1), que exibe características de cada um dos trabalhos. A escolha das características foi realizada em função do problema tratado pelo framework MaPS.

Os itens analisados para cada um dos trabalhos são os seguintes:

- *Utiliza um modelo de dados do usuário*: esta característica indica se o projeto utiliza informações de usuários em algumas de suas operações;

- *Considera informações de contexto*: esta característica aponta se os projetos utilizam algum tipo de informação de contexto a fim de otimizar seus processos;

- *Permite a utilização de diversos tipos de canais de comunicação*: este item indica se o projeto não restringe o uso de canais comunicação a um conjunto finito imutável, possibilitando que sejam acrescidas novas formas de interação quando necessário;

- *Oferece um mecanismo de busca por pessoas*: oferecer este mecanismo significa que o sistema possibilita que seus usuários possam realizar pesquisas por pessoas através de filtros, restringindo o seu resultado;

- *Leva em consideração os canais de comunicação na busca por pessoas*: esta característica indica se o projeto prevê a utilização do tipo de informação "canal de comunicação" como um fator limitante em seus resultados de pesquisa. Esta informação pode ser obtida através de um perfil, onde o usuário selecione as formas de interação que prefere, ou ainda, determinando-se que tipo de dispositivo o usuário carrega consigo, por exemplo;

- *Oferece uma biblioteca ou serviços destinados ao desenvolvimento de aplicações*: neste ponto são identificados projetos que também têm como objetivo oferecer este tipo de suporte para desenvolvedores, e não somente propor uma aplicação fechada;

- *Pode ser estendido para outros domínios ou tipos de aplicação*: este item informa se o trabalho foi projetado para que seja reutilizado em outros cenários que não prevê inicialmente;

Cada coluna da tabela representa um projeto, e o símbolo "x" em cada célula indica que o projeto correspondente contempla a característica informada na linha. O símbolo "o" na célula indica que o projeto expõe a característica como uma idéia, contudo não a especifica.

Tabela 3.1: Comparativo entre os projetos estudados

Característica	uLearning	UbiCollab	P2P e Redes Sociais	Collaborator
Utiliza um modelo de dados do usuário	x	x	x	x
Considera informações de contexto	x	x		x
Permite a utilização de diversos tipos de canais de comunicação	x	x		x
Oferece um mecanismo de busca por pessoas	o	o	x	
Leva em consideração os canais de comunicação na busca por pessoas	x			
Oferece uma biblioteca ou serviços destinados ao desenvolvimento de aplicações		x		x
Pode ser estendido para outros domínios ou tipos de aplicação	x	x	x	x

Dentre os projetos estudados, observa-se que todos usam um modelo de dados de usuários, porém, de maneiras diferentes. Para Collaborator, o perfil de usuário contém informações que auxiliam o sistema a identificar que tipo de dispositivo o usuário está portando, com o intuito de oferecer um conteúdo e serviços adaptados à sua situação. Já UbiCollab prevê um conjunto de meta-dados para suas entidades, incluindo-se usuários, contudo não especifica qual a função destes dados na aplicação. Diferentemente, o projeto de P2P e redes sociais usa perfis sociais dos usuários com o objetivo de montar sua rede social, que é usada para descrever as relações sociais entre os participantes. Enquanto isso, uLearning faz uso de informações contidas nos perfis dos usuários para otimizar a busca por pessoas dentro do ambiente, retornando resultados adaptados a seus perfis.

A maioria dos projetos, como pode ser observado na tabela, utiliza informações de contexto a fim de otimizar suas funcionalidades. No entanto, apenas uLearning usa estas informações especificamente no processo de busca, visto que os outros projetos que fazem uso de informações de contexto não determinam um processo semelhante.

Um suporte a diversos tipos de canais de comunicação também é oferecido pela maioria dos projetos, excetuando-se um que prevê o uso somente de IM (*Instant Messaging*). Entretanto, apenas uLearning considera este tipo de informação na busca por pessoas para uma colaboração. Contudo, a busca e sugestão de canais de comunicação mais indicados para uma interação é uma característica que não é observada em nenhum dos projetos estudados.

uLearning é um projeto que, mesmo apresentando a importância do processo de busca por pessoas para uma colaboração, não o especifica suficientemente, mesmo em seus trabalhos mais recentes (Zhang and Jin, 2005; Zhang et al., 2005b,a), de modo que não é

possível afirmar, por exemplo, como informações de contexto ou mesmo o perfil de seus usuários são levados em consideração nas pesquisas por colaboradores. Assim como Ubi-Collab, que também enfatiza a busca por pessoas e seleção de canais, mas não descreve como é feito este processo e nem indica em sua arquitetura quais seriam os componentes utilizados para tal funcionalidade. O projeto de P2P, entretanto, apresenta o mecanismo de busca de pessoas em função de seus perfis.

Ao analisar a penúltima característica apresentada na tabela comparativa, nota-se que dois projetos têm como preocupação o oferecimento de suporte para desenvolvedores, possibilitando a criação de novas aplicações, e não restringindo a um domínio somente.

Observando a última característica estudada nos projeto, percebe-se que todos os projetos providenciam uma compatibilidade mínima com aplicações de outros domínios de aplicações colaborativas. O que torna clara a idéia de reutilização em diversas aplicações.

3.6 Considerações sobre o Capítulo

O capítulo 3 apresenta um conjunto de propostas existentes para suporte à colaboração e identifica suas principais características, limitações e contribuições.

O próximo capítulo apresenta o framework MaPS, uma nova alternativa para tratar o problema de seleção de pessoas e canais de comunicação no processo de colaboração que busca contornar as limitações das propostas estudadas.

4 O FRAMEWORK MAPS

Este capítulo introduz o framework MaPS (*Matching People to Share*), o qual apresenta-se como uma alternativa para construção de sistemas colaborativos em ambientes ubíquos. O framework busca tratar as limitações das propostas existentes, identificadas no capítulo 3. O capítulo inicia-se pela caracterização do ambiente alvo do framework, identificando-se as funcionalidades complementares que serão assumidas como disponíveis pelo próprio ambiente. A seguir são identificados os requisitos funcionais e não funcionais a serem atendidos pelo MaPS. Por fim, a proposta de framework é apresentada e detalhada a partir de duas perspectivas: funcional e arquitetural.

4.1 Requisitos de Ambiente

Uma das motivações do MaPS é a definição de uma solução que viabilize a sua utilização em diferentes cenários de colaboração em ambientes ubíquos. Contudo, existem requisitos básicos que devem ser atendidos pelo ambiente no qual o sistema que utiliza o framework será executado. Estes requisitos compreendem funcionalidades não cobertas pelo framework, mas que são necessárias a sua operação.

Os requisitos a serem supridos pelo ambiente estão organizados em três serviços:

- *Serviço de gerenciamento de perfis de pessoas*. Este serviço é responsável pelo cadastro, atualização, persistência e disponibilização dos perfis dos usuários no ambiente. Opcionalmente, este serviço pode ainda manter um histórico de colaborações relacionadas a um perfil de usuário;

- *Serviço de contexto.* Este serviço disponibiliza informações sobre contexto dos usuários e dos recursos, pertinentes à colaboração. Freqüentemente, o tipo de informação de contexto mais relevante para colaboração reflete aspectos de mais alto nível, portanto, o requisito para este tipo de serviço é também prover dados interpretados de contexto;

- *Serviço de base de dados lexicais*. Esta estrutura mantém uma base de termos, relacionados semanticamente, utilizada para estabelecer a equivalência ou similaridade

semântica entre termos empregados nos processos de busca.

MaPS não requer um modelo específico de perfil de usuários, contudo há um conjunto de informações básicas que devem ser contempladas pelo perfil utilizado pela aplicação. As informações necessárias para o bom funcionamento do framework estão listadas na tabela 4.1.

Tabela 4.1: Informações básicas para o perfil do usuário

Informação	Descrição
Nome	Nome/identificação pelo qual o usuário será reconhecido no sistema por outros participantes
Identificador	Identificador único, de uso da aplicação, que é utilizado para diferenciar os usuários
Conhecimentos	Lista de conhecimentos que o usuário tem
Canais de comunicação	Lista de canais de comunicação que o usuário possui
Preferência de canais de comunicação	Lista de canais de comunicação que o usuário tem preferência em usar

Ressalta-se aqui que o dado de canal de comunicação deve indicar ao menos que tipo de canal está sendo tratado (por exemplo, e-mail ou telefone) e qual a forma de comunicação utilizada (síncrona ou assíncrona).

A aplicação que utilizar o MaPS deve possuir informações equivalentes a estas apresentadas. Adicionalmente às informações básicas, o serviço de gerenciamento de perfis pode disponibilizar informação histórica sobre as colaborações que já ocorreram no ambiente, relacionando-as com os perfis envolvidos. Tais informações, quando disponíveis, pode ser aproveitadas na busca de perfis realizada pelo MaPS.

As informações de contexto utilizadas pelo framework são definidas em função do tipo de aplicação que faz uso do MaPS. Por exemplo, para uma aplicação que almeje como resultado final o encontro físico de usuários colaboradores, a localização é um dado importante no processo de busca por pessoas. Já uma aplicação que visa somente a interação virtual entre os participantes, localização não é um item vital a ser levado em consideração. Sendo assim, não é definido um conjunto mínimo de dados que o serviço de contexto deve prover. Cabe ao desenvolvedor determinar a lista de informações necessárias e com isso, escolher um serviço de contexto que atenda às suas expectativas. Contudo, MaPS define um conjunto de diretivas que devem ser observadas ao se disponibilizar as informações de contexto. Para cada elemento de contexto obtido (um valor de contexto relacionado a uma entidade), este deve prover os seguintes dados apresentados na tabela 4.2.

Sobre estas diretivas, o valor absoluto corresponde a dados de alto nível, ou seja, informações interpretadas de contexto. A confiabilidade diz respeito ao quanto de certeza tem-se em relação à informação que está sendo passada, visto que, uma informação de

Tabela 4.2: Diretivas para informações de contexto

Informação	Descrição
Value	O valor absoluto da informação
Confidence	A confiabilidade desta informação
Entity	O identificador da entidade relacionada a este valor
Timestamp	A data/tempo em que esta informação foi obtida
Type	O tipo de informação que está sendo passada

Value	35
Confidence	85
Entity	_room0382
Timestamp	Tue Oct 21 14:16:57 2008
Type	locate.room.temperature

Figura 4.1: Exemplo de informação de contexto

contexto pode ser deduzida de um conjunto de fatores. Neste caso, há uma chance de acerto desta informação, isto é expresso pela confiabilidade.

O identificador da entidade é único, pois é a forma de localizá-la em um ambiente. E o tipo de dado relativo a um valor de contexto especifica um atributo da entidade da qual se está tratando. Por exemplo: sobre a entidade "sala", um tipo de informação de contexto é a "temperatura".

Um exemplo de informação de contexto válida para o MaPS é a apresentada na figura 4.1.

O próximo serviço externo, a base de dados lexicais, pode ser descrito como um repositório de informações que englobam não só descrições estáticas dos termos, mas também suas propriedades e valores associados (Ooi, 1998) (através de relações semânticas como hipernímia e hiponímia, por exemplo).

O serviço de base de dados lexicais, assim como o serviço de contexto, é um elemento que varia de aplicação para aplicação. Seu maior objetivo é auxiliar nas buscas por colaboradores, expandindo as possibilidades de resultados de uma requisição. Esta expansão consiste em procurar palavras semanticamente semelhantes àquelas informadas como critério de busca pelo usuário, de forma a abranger um número maior de resultados. Portanto, o domínio desta base de dados deve estar de acordo com domínio da aplicação. Contudo, como será visto posteriormente, somente o uso da expansão de critérios de busca não é garantia de uma resposta mais satisfatória. Idealmente, a aplicação deve utilizar este mecanismo somente em alguns casos, discernindo as melhores situações.

4.2 Requisitos para o Framework

A figura 4.2 apresenta um cenário de colaboração ubíqua, onde são identificadas as principais ações tomadas pelos atores envolvidos.

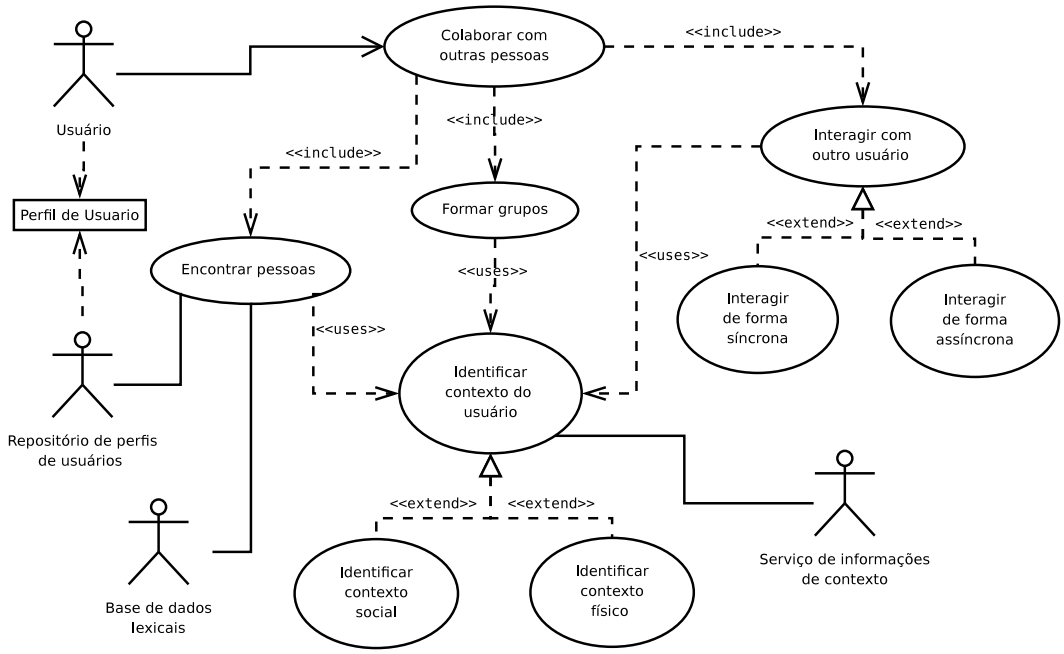


Figura 4.2: Cenário de colaboração ubíqua

Abaixo, estão descritos os principais casos de usos mostrados no diagrama da figura 4.2.

1. *Colaborar com outras pessoas.* O processo de colaboração na verdade acaba sendo um agregador de outras ações: encontrar pessoas, interagir com elas ou mesmo formar grupos (as três etapas da colaboração já apresentadas). Nos casos de uso posteriores estes passos são especificados.

2. *Encontrar pessoas.* Primeiro passo de uma colaboração, esta é uma ação que envolve uma pesquisa. Nesta pesquisa, o usuário seleciona pessoas em função de critérios pré-estabelecidos. Para encontrar pessoas dados como perfis de usuários, informações de contexto e dados lexicais podem ser consultados.

3. *Interagir com outro usuário.* A interação com outra pessoa pode se dar de maneiras distintas. Seja sincronamente (através de contato pessoal, uso de clientes de mensagens instantâneas ou mesmo através de vídeo conferência) ou assincronamente (utilizando e-mails, fóruns ou wikis), esta é uma característica fundamental da colaboração.

4. *Identificar contexto do usuário.* Um serviço externo deve prover informações do contexto para que possam ser utilizadas na seleção de pessoas (na busca por colaboradores) e de canais de comunicação (na sugestão de mecanismos de interação).

5. *Formar grupos.* A formação de grupo é uma prática comum em um ambiente de colaboração. Esta ação, geralmente, é antecedida de uma fase de busca e, posteriormente, há a interação entre os componentes do grupo.

Entende-se que todos estes aspectos sejam importantes no processo de colaboração, entretanto MaPS atua especificamente no caso de uso 2 (*Encontrar pessoas*), através da solução de um mecanismo de busca de usuários; e indiretamente no caso de uso 3 (*Interagir com outro usuário*), ao indicar e propor canais de comunicação a serem utilizados pelo usuário em sua interação. Além disso, o framework faz uso também do caso de uso 4 (*Identificar contexto do usuário*) a fim de obter informações de contexto dos usuários.

Foram definidos também requisitos não funcionais para o framework:

- *Ser facilmente extensível*. Com esta característica, é possível atender a diversos cenários, pois o framework é maleável e seus conceitos podem ser adaptados de acordo com o ambiente no qual será utilizado. Deste modo, é útil para vários tipos de aplicações;

- *Suporte a um modelo flexível de perfil de usuários*. Entende-se que nem todas as aplicações necessitam de um mesmo modelo de perfil de usuários. Por isso, o framework deve permitir que o desenvolvedor trabalhe com o modelo de perfil que mais se adapte à utilização do framework em suas aplicações, não restringindo a escolha a um grupo de modelos pré-definidos;

- *Suporte a diferentes heurísticas de busca por pessoas*. Nem toda busca é realizada com o mesmo número de informações. Deve ser possível utilizar diferentes algoritmos de busca, para serem aplicados em função das necessidades da aplicação.

- *Permitir a escolha das aplicações de serviços externos.* Como descrito na seção 4.1, MaPS utiliza um conjunto de serviços externos. Contudo, o framework não deve definir especificamente quais aplicações de serviço deve usar. Assim, MaPS deve ser projetado de modo que faça uso de qualquer aplicação de serviço que preencha os requisitos informados na seção 4.1.

4.3 O Framework MaPS

O framework MaPS será apresentado em duas partes: uma visão funcional e uma visão arquitetural. A visão funcional, abordada na seção 4.3.1, mostra os conceitos básicos do

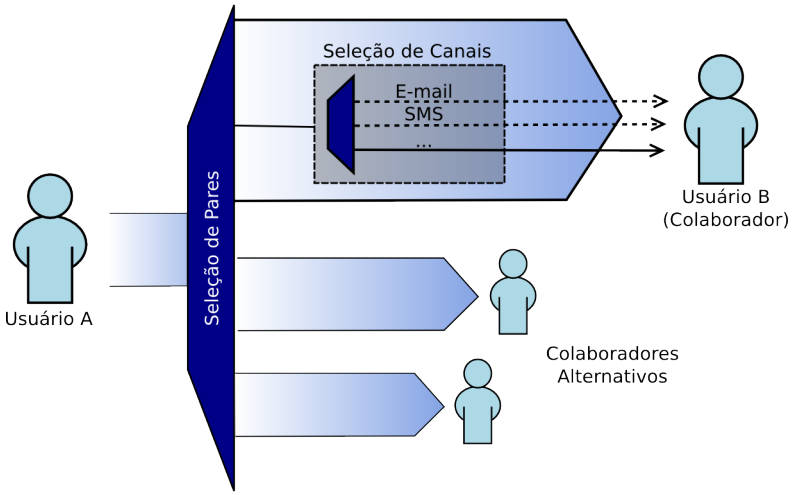


Figura 4.3: Passos para colaboração

framework e seus relacionamentos. A visão arquitetural, discutida na seção 4.3.2, enfoca os aspectos de instanciação e organização dos componentes do MaPS.

MaPS foi definido seguindo a metodologia de desenvolvimento dirigido por *hot-spot* (Pree, 1994).

4.3.1 Aspectos Funcionais

Como visto na seção 2.1, o processo de colaboração envolve dois passos básicos: a busca por pessoas e após, a definição do canal de comunicação a ser utilizado para a interação entre as pessoas (ver figura 4.3). MaPS incrementa o processo de colaboração, auxiliando o desenvolvimento de aplicações através da tomada de decisão nestes dois pontos críticos. O uso do MaPS automatiza e otimiza assim, as etapas de seleção de pares e de canais de comunicação, sugerindo as opções mais adequadas para o usuário em função do ambiente (contexto) em que ele se encontra.

Tanto na seleção de pares, quanto na seleção de canais de comunicação, elementos do contexto de todos usuários envolvidos podem ser levados em consideração, refinando-se a consulta. Neste caso, é importante ressaltar que a seleção destes elementos vai ser influenciada pelo cenário no qual o framework está instanciado.

4.3.2 Arquitetura

Os conceitos utilizados para construir as duas funcionalidades do MaPS (escolha de pessoas e de canais de comunicação) estão exibidos na figura 4.4, que representa a macro arquitetura do framework.

A construção da seleção de pares é feita, basicamente, através dos conceitos de (i) perfis que contêm informações de usuários (*Profile*), (ii) *matcher* que realiza o casamento de perfis (*ProfileMatcher*), e (iii) critérios de *matching* que usam operadores predefinidos para especificar a busca (*Criterion*).

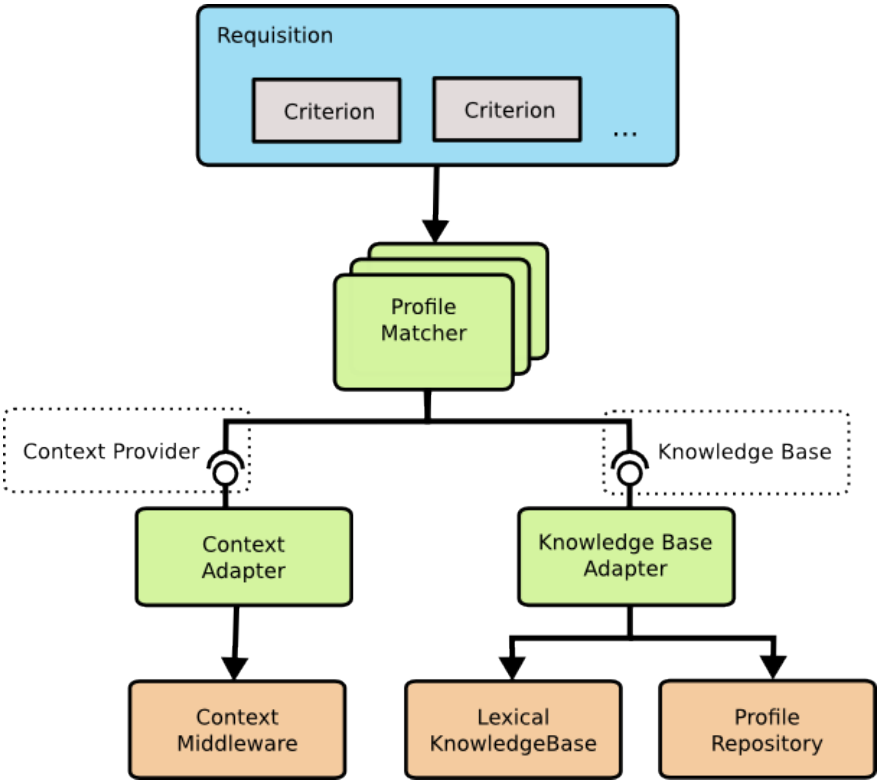


Figura 4.4: Macro arquitetura do MaPS

Como descrito na seção 4.1, o ambiente deve prover um serviço de perfis de usuários. Estas informações estão representadas no elemento *Profile*, que é disponibilizado através do serviço externo de repositório de perfis (*Profile Repository*). Cada usuário existente no ambiente é identificado através de um *Profile*.

ProfileMatcher é o elemento que faz o casamento de *Profiles*. Ele utiliza-se de informações provenientes da *KnowledgeBase*, que representa uma abstração de repositório persistente que centraliza as informações pertinentes aos processos de colaboração, e ainda informações obtidas através do *ContextProvider*. Estas informações estão definidas na fase de identificação dos requisitos funcionais do ambiente colaborativo, e correspondem aos dados de usuários, base de dados lexicais e dados do contexto (vide seção 4.1). O acesso do framework a serviços externos se dá através de uma interface (*Context Provider* e *Knowledge Base*) e um adaptador (*Context Adapter* e *Knowledge Base Adapter*), seguindo a idéia do padrão de projeto *Adapter* (Gamma et al., 2000).

O elemento *ProfileMatcher* está construído sob a visão do padrão de projeto *Strategy* (Gamma et al., 2000), onde define uma interface comum para todas as heurísticas de busca de perfis. A esquematização desta organização pode ser observada na figura 4.5.

SimpleMatcher e *LocationAwareMatcher*, ao implementar a interface *ProfileMatcher*, disponibilizam estratégias concretas para a busca por perfis. A relação entre o elemento *ProfileMatcher* e suas implementações está marcada ainda com o esteriótipo *incomplete*, da UML-F (Fontoura et al., 2000). Este indica que se trata de um *hot-spot*, e neste caso

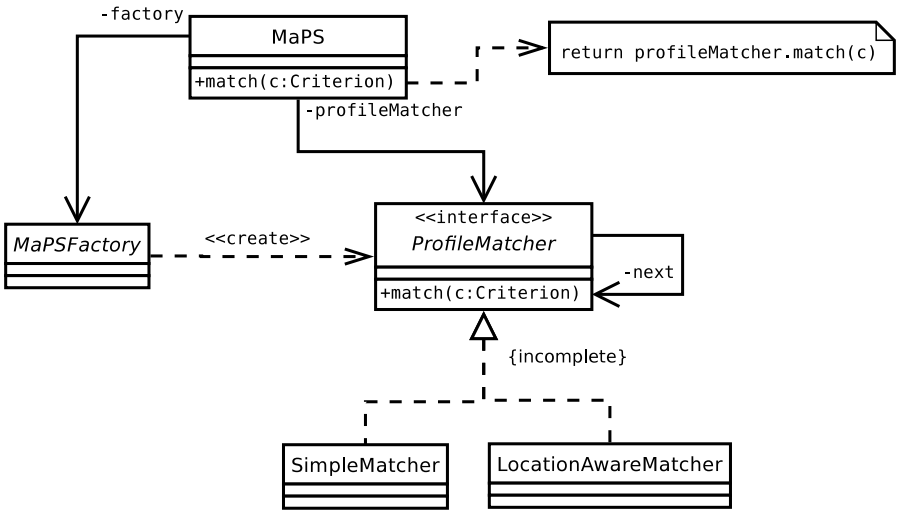


Figura 4.5: MaPS ProfileMatcher - Diagrama de classes

sinaliza que permite que novas subclasses, com diferentes algoritmos de busca, sejam definidas na instanciação do framework. O porquê de possibilitar haver mais de um algoritmo para realizar uma mesma função deve-se aos seguintes fatos:

- Nem todos os cenários necessitam das mesmas informações para realizar uma busca.
- Não é eficiente ter um algoritmo que contemple todas as opções possíveis de diferentes cenários.
- Nem sempre todas as informações que desejamos estão disponíveis. Portanto, com base nas informações que se tem, pode-se selecionar o algoritmo que melhor se adeque à situação atual.

Cada especialização da classe *ProfileMatcher* é responsável por implementar uma heurística distinta para a funcionalidade, sendo a utilização de uma heurística específica determinada por uma configuração do framework ou pelo estado do contexto. Por exemplo, se houver uma informação de localização de usuários, o algoritmo pode levar este dado em consideração na hora de selecionar as pessoas. A forma de seleção do algoritmo a ser usado é feita através da instanciação do padrão de projeto *Abstract Factory* (Gamma et al., 2000). Neste caso, ainda observando a figura 4.5, nota-se que a classe *MaPS* tem o papel de *Client*, *MaPSFactory* atua como *AbstractFactory*, *ProfileMatcher* representa um *AbstractProduct*, enquanto que as classes *SimpleMatcher* e *LocationAwareMatcher* são tratados como *Products*.

SimpleMacther representa uma busca por colaboradores em função de atributos de seus perfis. Tem como principal objetivo o tratamento do operador LIKE e dos critérios opcionais, para então consultar informações da KnowledgeBase. Já LocationAwareMatcher trabalha também com informações de contexto, acessando para tal, o serviço externo de contexto.

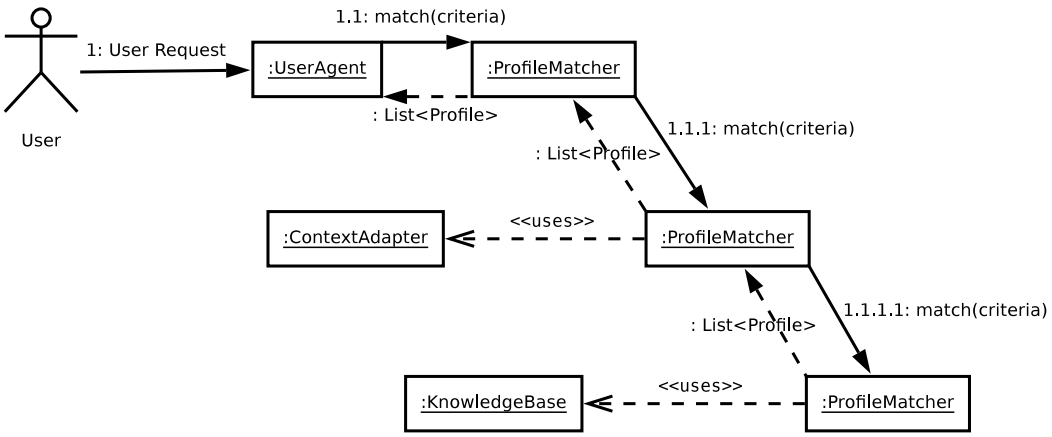


Figura 4.6: *Pipeline* de Filtros

O framework permite que os elementos *ProfileMatcher* possam ser estruturados como um *pipeline*, fazendo com que a busca por *Profiles* possa ser refinada por diversos filtros de forma incremental. A figura 4.6 ilustra esta idéia. Nela, uma requisição de alto nível feita pelo usuário é mapeada pelo *UserAgent* (uma entidade genericamente citada aqui, que representa um agente que está com o usuário e intermedeia as requisições realizadas pelo usuário para a aplicação servidor) para uma busca por pessoas, que obedece a um determinado conjunto de critérios. Este conjunto de critérios é transportado através do *pipeline*, podendo ser refinado pela heurística implementada em cada estágio. Ao atingir o estágio final, a implementação de *ProfileMatcher* acessa a *KnowledgeBase* retornando uma lista preliminar de usuários que satisfazem aquele conjunto potencialmente expandido de critérios. A lista flui na direção oposta do *pipeline*, podendo ser filtrada novamente a cada estágio visando melhorar a qualidade do resultado da busca. Por exemplo, um elemento intermediário do *pipeline* pode classificar os elementos da lista em função de um atributo de contexto, como localização.

A busca por *Profiles* é sempre realizada em função de critérios, que podem ser simples ou compostos. Critérios simples são aqueles que relacionam pares de "atributo-valor" através de um operador e que devem ser satisfeitos em uma consulta. Por outro lado, critérios compostos representam um grupo de critérios aninhados de forma a especializar uma consulta. Para capturar esta idéia, MaPS instancia o padrão de projeto *Composite* (Gamma et al., 2000), onde a interface *Criterion* representa um critério em sua forma abstrata, enquanto *SimpleCriterion* e *CompositeCriterion* representam, respectivamente, critérios simples e compostos. Tal organização é ilustrada na figura 4.7.

O elemento *SimpleCriterion* faz uso ainda de operadores de valor. Estes operadores atuam sobre os valores especificados em cada critério. Abaixo estão descritos os operadores utilizados:

- *CONTAINS*. Indica que as pessoas retornadas na consulta possuem o valor especificado no critério;

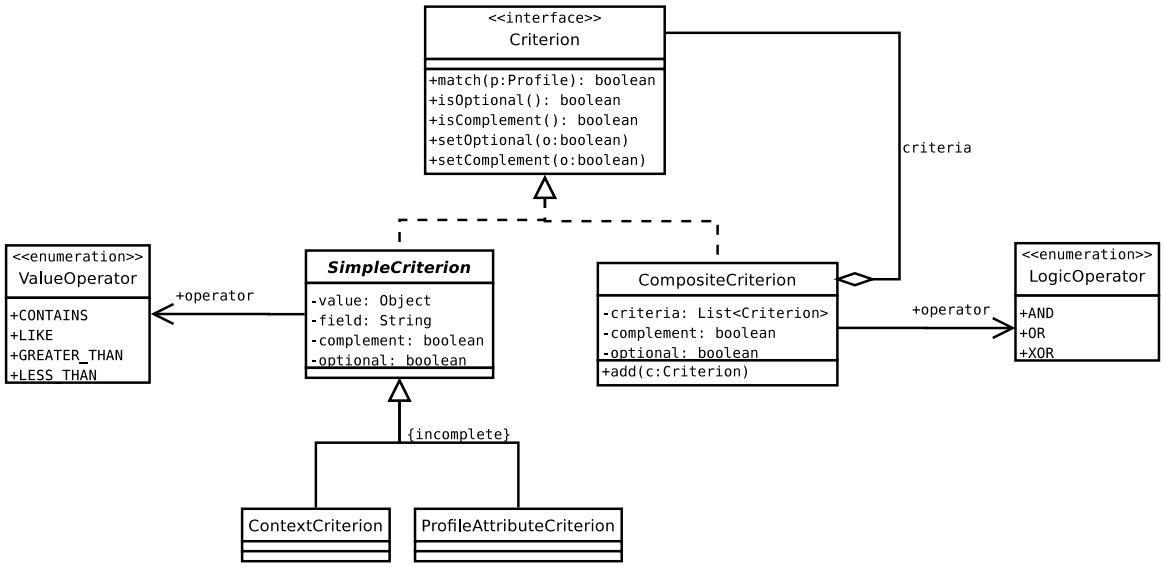


Figura 4.7: MaPS *Criterion* - Diagrama de classes

- *LIKE*. Mostra no resultado da consulta pessoas que possuem ou o valor especificado no critério ou um valor aproximado. Este valor aproximado é disponibilizado por um serviço externo: a base de dados lexicais, como citado na seção 4.1;
- *GREATER_THAN*. Indica que as pessoas que retornarem na consulta possuem um valor maior do que o informado no critério;
- *LESS_THAN*. Significa que as pessoas que retornarem na consulta possuem um valor menor do que o especificado no critério.

Em *CompositeCriterion*, um grupo de critérios já criados são organizados em função de três operadores lógicos:

- *AND*. Indica que todas as condições devem ser atendidas.
- *OR*. Define que pelo menos uma das condições deve ser atendida.
- *XOR*. Determina que apenas uma das condições deve ser atendida.

Além dos operadores de valor e lógicos, os elementos *SimpleCriterion* e *Composite-Criterion* possuem duas *flags*:

- *complement*. Inverte a aplicação do critério, ou seja, indica que o critério é considerado satisfeito caso o registro a ser verificado não atenda à restrição especificada no critério.
- *optional*. Indica que este é um critério opcional. A não satisfação de um critério opcional não implica a desclassificação do registro. Por outro lado, na comparação entre dois registros do conjunto de resultados, um que atenda a este tipo critério é considerado de melhor qualidade que outro registro que não o faça.

SimpleCriterion possui ainda duas extensões que diferenciam critérios em função da procedência dos valores a serem pesquisados. Enquanto que critérios criados através do elemento *ProfileAttributeCriterion* referenciam informações da *KnowledgeBase*, elementos *ContextCriterion* implicam interagir com o *ContextProvider* para obtenção de informação de contexto.

A ligação do framework com o serviço de contexto provido pelo ambiente (que fornece as informações de contexto necessárias para a busca por pessoas) dá-se através de adaptadores que implementam a interface *ContextProvider*. Estes adaptadores são responsáveis por converter as demandas geradas pelo MaPS em chamadas ao *middleware* de contexto subjacente. As informações obtidas junto ao *middleware* são convertidas em um formato padrão, caracterizado por quatro atributos, como descrito na tabela 4.2.

A implementação de *ContextProvider* deve possibilitar ao framework obter informações sobre o estado do contexto de usuários. Pode-se esperar que as informações de contexto pertinentes a operação do MaPS pertençam, tipicamente, a um de dois grandes grupos: descrevam aspectos do ambiente físico do usuário ou caracterizem circunstâncias sociais nas quais este está imerso. Além disso, tais consultas sobre o contexto podem tanto trabalhar com valores absolutos quanto com valores relativos. Por exemplo, pode-se questionar sobre a localização física exata de um usuário (contexto absoluto) ou então sua localização relativa (perto, longe) a um outro usuário (contexto relativo). Outra informação típica de contexto que deve ser disponibilizada pelo *ContextProvider* diz respeito à informação de presença (*presence information*), que representa um dado publicado por um usuário para outros participantes com o objetivo de indicar a sua disponibilidade (Brok et al., 2006) (por exemplo, *online*, *offline* ou ocupado).

Complementando a integração do MaPS com o ambiente de instalação da aplicação, a interface *KnowledgeBase* realiza a ponte entre o framework e os serviços de repositório de perfis e da base de dados lexicais. A *KnowledgeBase* inclui um mecanismo elementar de busca sobre o qual o *ProfileMatcher* constrói uma busca mais qualificada considerando critérios de pesquisa de mais alto nível (por exemplo, operador LIKE, informação de contexto).

As informação disponibilizadas na *KnowledgeBase* podem ser de três naturezas:

- *Perfis*. Dados de perfis de usuários como conhecimentos, canais de comunicação disponível, etc.

- *Termos*. Conjunto de palavras, organizados semanticamente. Em particular, são do interesse do MaPS relacionamentos do tipo hipernímia, hiponímia e sinônimos, pois permitem flexibilizar os critérios de busca.

- *Histórico de colaboração*. Cada registro presente no histórico descreve uma situação de colaboração que ocorrei entre dois usuários da plataforma. Tal registro pode

incluir dados como data, assunto (conhecimento envolvido), meio de comunicação utilizado, etc.

Na seção a seguir é detalhada a implementação do MaPS e exemplos de utilização.

4.4 Protótipo

O protótipo do MaPS foi desenvolvido utilizando-se tecnologia Java[1], por ser uma linguagem robusta, orientada a objetos e de grande portabilidade. Com isso, as possibilidades de utilização do framework pelos mais diversos tipos de aplicações são diversas. No desenvolvimento do protótipo utilizou-se a versão 1.6 do JDK do Java e a IDE selecionada foi Eclipse[2].

O diagrama da figura 4.8 apresenta o diagrama de classes principais do MaPS.

A classe MaPS inspira-se no padrão de projeto *Façade* (Gamma et al., 2000), fornecendo um ponto unificado de acesso à funcionalidade do framework para as aplicações. Além de repassar requisições de busca (através do método *match*) para a instância da classe *ProfileMatcher*, esta classe contém referências para as instâncias dos principais elementos utilizados no processo de busca: *ProfileMatcher*, *KnowledgeBase* e *ContextProvider*. As referências para estes objetos são obtidas através dos métodos *getProfileMatcherChain*, *getKnowledgeBase* e *getContextProvider*, respectivamente, que acessam *MaPSFactory*.

MaPSFactory atua na configuração da instanciação do framework. Para tal, faz uso do arquivo de configuração *MaPS.properties*. Seu conteúdo é mostrado na figura 4.9.

MaPS possui duas modalidades principais, onde pode atuar localmente ou sob uma organização cliente-servidor. A principal diferença entre estes dois modos está em permitir que se possa construir um pipeline de *ProfileMatcher* de modo que fique parte no cliente e parte no servidor (idéia mostrada anteriormente, na figura 4.6). Esta configuração é definida utilizando a propriedade "*maps.factory*" do arquivo de configuração, onde indica-se qual classe *factory* deve ser utilizada. Para um funcionamento local, esta propriedade deve utilizar a classe *DefaultFactory*. Para um funcionamento distribuído, devem ser observados os seguintes pontos:

- tanto cliente quanto servidor devem possuir uma instância do framework MaPS;
- a aplicação cliente deve utilizar a classe *ClientFactory*;
- a aplicação servidor deve utilizar a classe *ServerFactory*.

[1] http://java.sun.com
[2] http://www.eclipse.org

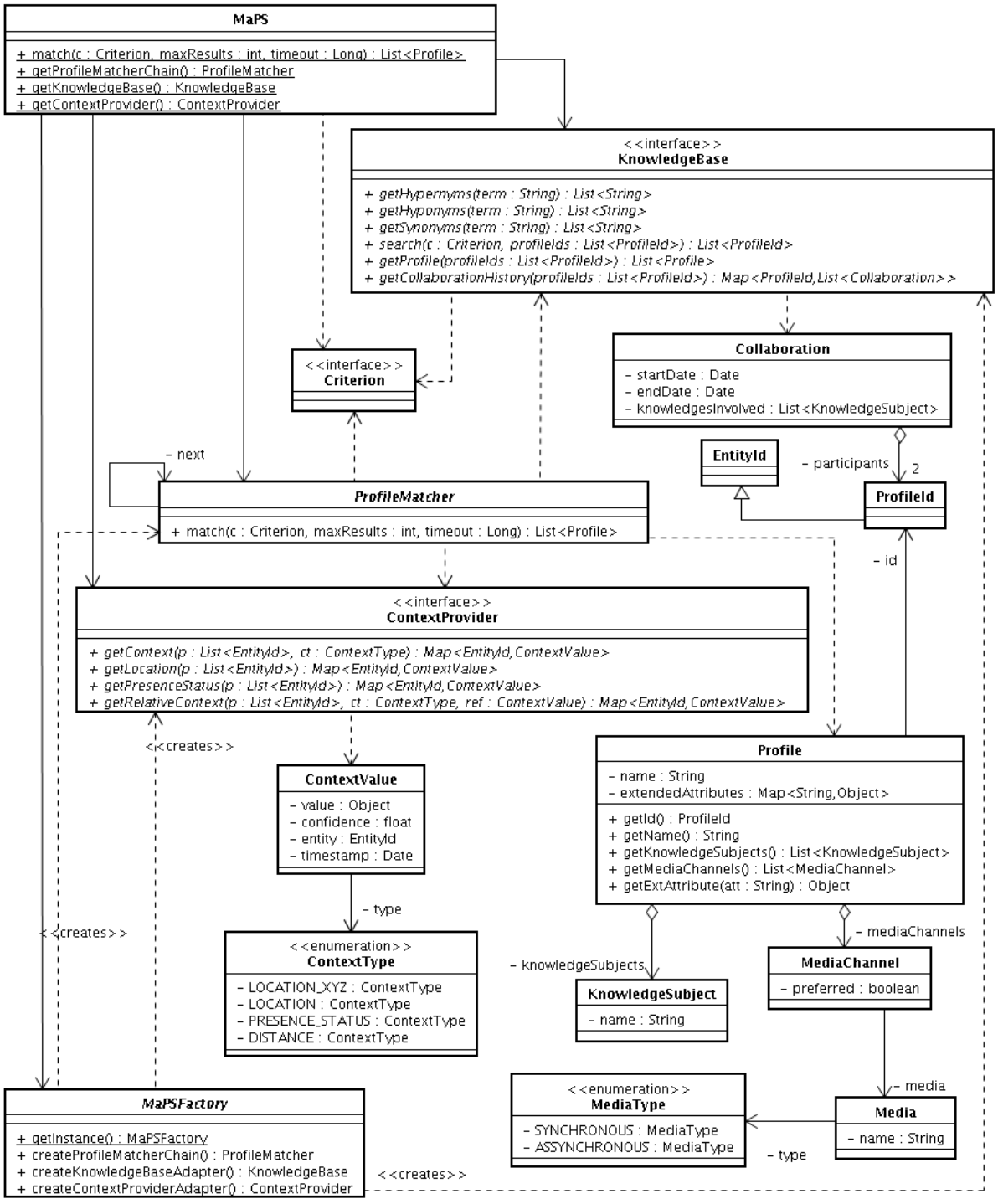


Figura 4.8: MaPS - Diagrama de classes

```
# MaPS - Arquivo de configuracao
#Habilita informacoes de debug na execucao do framework
#maps.debug=true
maps.debug=false

#Factory da instancia do MaPS
maps.factory=maps.factory.DefaultFactory
#maps.factory=maps.factory.ClientFactory
#maps.factory=maps.factory.ServerFactory

## Opcoes para ProfileMatcher
#Chain
maps.profileMatcher.0=maps.matcher.LocationAwareMatcher
maps.profileMatcher.1=maps.matcher.SimpleMatcher
#URL do servidor para acesso do cliente ProfileMatcher
maps.remoteServer=10.0.0.1

## Opcoes para ContextAdapter
maps.contextAdapter=

## Opcoes para KnowledgeBase
maps.knowledgeBaseAdapter=
```

Figura 4.9: Arquivo de configuração - MaPS

ClientFactory tem como objetivo acrescentar como último elemento do pipeline de instâncias de *ProfileMatcher* um stub para se comunicar com o servidor, e assim, completar a requisição feita pelo cliente. Já *ServerFactory* inicia o pipeline de *ProfileMatcher* com uma requisição externa, vinda de um cliente.

A configuração do pipeline de instâncias de *ProfileMatcher* também é feita através do arquivo de configuração. A classe *MaPSFactory*, através de seu método *createProfileMatcherChain* monta a seqüência de instâncias em função das propriedades "*maps.profileMatcher*", seguindo a ordem estipulada no arquivo de configuração. Os valores válidos para esta propriedade são as classes que estendem *ProfileMatcher* (uma classe abstrata, como visto anteriormente no diagrama da figura 4.5). O framework já disponibiliza duas extensões que implementam diferentes algoritmos: *SimpleMatcher* e *LocationAwareMatcher*.

Outra informação ainda que deve constar no arquivo de configuração do framework são os indicadores de classes *adapter*, que implementam as interfaces de acesso aos serviços externos (*KnowledgeBase* e *ContextProvider*).

A construção de critérios compostos pode tornar-se uma tarefa repetitiva e, portanto, sujeita a erros, devido à complexidade dos encadeamentos que podem ser construídos. Para facilitar esta tarefa MaPS oferece na package *maps.util* a classe *CriteriaBuilder* que serve de atalho às operações de criação e composição de critérios, como pode ser observado na figura 4.10. As funções existentes na classe *CriteriaBuilder* representam todas as opções de operadores lógicos e de valor que atuam sobre critérios que dizem respeito

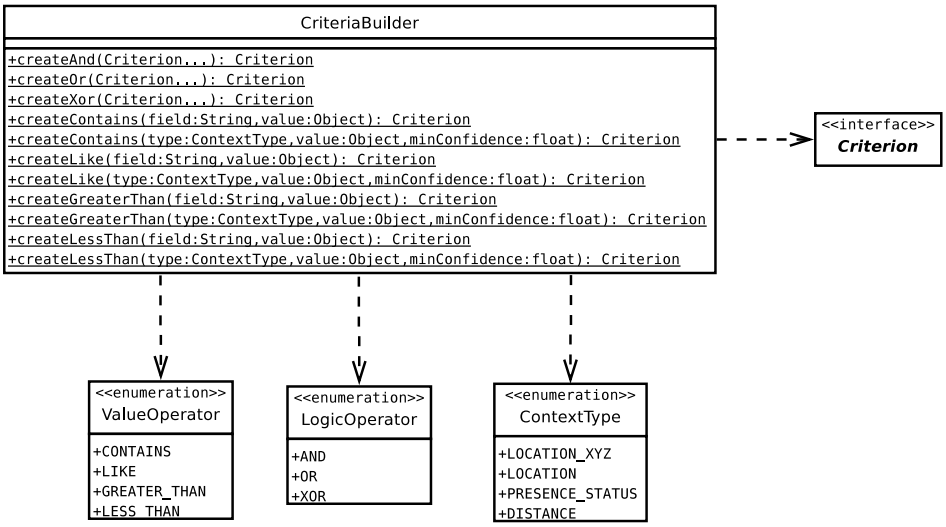


Figura 4.10: Classe utilitária *CriteriaBuilder*

a informações de perfil (*ProfileAttributeCriterion*) ou de contexto (*ContextCriterion*).

É importante notar que esta classe é um utilitário para facilitar a criação de critérios já previstos no framework. Contudo, vale ressaltar que *SimpleCriterion* pode ser estendida para incorporar novos tipos de critérios ao framework, e para tanto, não há um utilitário de criação. Neste caso, a criação de critérios personalizados deve ser feita da maneira tradicional (instanciando as classes necessárias).

A classe *Criterion* é a representação do elemento critério que será utilizados na busca por pessoas (seu detalhamento pode ser observado no diagrama da figura 4.7). A seleção por estes critérios pode ser feita de forma explícita pelo usuário ou por algum mecanismo automático do sistema.

Para um melhor entendimento dos operadores utilizados nos critérios, são exibidos fragmentos de código Java que exemplificam a criação de critérios.

No primeiro exemplo (figura 4.11), é feita uma consulta por pessoas que se chamam "Pedro", ou seja, cria-se um critério indicando que o campo de pesquisa é *ProfileAttr.NAME* (a classe *ProfileAttr* contém constantes dos principais atributos de perfil) e o valor de pesquisa é "Pedro". Neste caso, deseja-se que o valor indicado como critério seja exato, portanto o operador a ser utilizado é o CONTAINS.

```
1 Criterion c = CriteriaBuilder.createContains(ProfileAttr.NAME, "
    Pedro");
```

Figura 4.11: Busca de pessoas por valores

No segundo exemplo (figura 4.12), deseja-se procurar pessoas que possuam conhecimentos em "filosofia oriental". O usuário não especifica exatamente que tipo de elemento da filosofia oriental deseja, portanto, queremos uma busca por termos equivalentes. Para realizar uma busca por equivalência, é comum utilizar termos com um significado mais

amplo, como apresentado.

```
1 Criterion c = CriteriaBuilder.createLike(ProfileAttr.KNOWLEDGE, "
    Filosofia Oriental");
```

Figura 4.12: Busca de pessoas por termos equivalentes

Esta equivalência de termos é dada através da base de dados lexicais. Através de uma pesquisa nesta base, é possível obter valores equivalentes e então, realizar uma busca entre as pessoas.

Para realizar uma pesquisa com mais de um filtro é utilizada a classe *CompositeCriterion*. Esta cria uma estrutura semelhante a uma árvore, onde as condições de pesquisa são agrupadas. Como exemplo, considere a seguinte situação: "*deseja-se pesquisar pessoas que conheçam Orientação a Objetos e que tenham conhecimentos na linguagem Java ou .NET*". Neste caso, a utilização das classes de critérios é mostrada na figura 4.13.

```
1 Criterion c1 = CriteriaBuilder.createOr(
2                    CriteriaBuilder.createContains(ProfileAttr.
                        KNOWLEDGE, "Java"),
3                    CriteriaBuilder.createContains(ProfileAttr.
                        KNOWLEDGE, ".NET"));
4 Criterion c2 = CriteriaBuilder.createAnd(
5                    CriteriaBuilder.createContains(ProfileAttr.
                        KNOWLEDGE, "Orientação a Objetos"),
6                    c1);
```

Figura 4.13: Busca composta

A consulta desejada pode ser construída em dois passos. Primeiro a criação de um critério composto através do método *CriteriaBuilder.createOr*, na linha 1, e suas respectivas adições de critérios (linhas 2 e 3), relacionando-os com este operador lógico. Segundo, a criação do critério composto com o operador lógico AND através do método *CriteriaBuilder.createAnd* (linha 4), a após, a inclusão as sentenças relacionadas por este operador (linhas 5 e 6).

Esta mesma consulta pode ser visualizada em forma de árvore, como exemplificado na figura 4.14.

No próximo exemplo (figura 4.15), o usuário deseja encontrar pessoas que estejam no auditório 205. Neste caso, estamos utilizando uma informação de contexto: localização (*ContextType.LOCATION*).

LocationId, um subtipo de *EntityId*, é utilizado para referenciar localizações simbólica no ambiente de forma única (por exemplo, uma sala, um prédio ou uma rua). Numa busca que envolva critérios que correspondem a informações de contexto, o tipo de critério utilizado é o *ContextCriterion*. Neste caso, devemos indicar o tipo de informação tratada

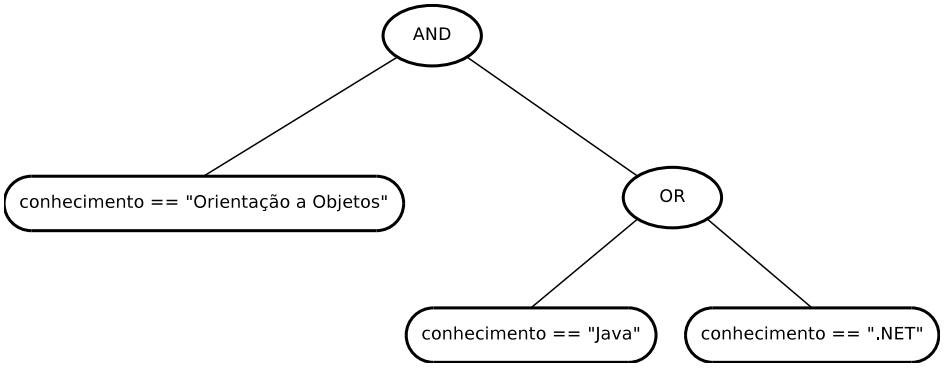


Figura 4.14: Árvore de busca - *CompositeCriterion*

```
1 LocationId sala205 = //obtém o identificador correspondente ao
      auditório 205
2 Criterion c = CriteriaBuilder.createContains(ContextType.LOCATION,
      sala205, 90);
```

Figura 4.15: Elemento de contexto como critério de busca de pessoas

(*ContextType.LOCATION)* e também o nível de confiabilidade mínimo que se deseja para tal informação.

Exemplos de utilização de critérios de busca por canais de comunicação pode ser visto nas figuras 4.16 e 4.17:

```
1 Criterion c = CriteriaBuilder.createContains(ProfileAttr.MEDIA, new
      Media(null, MediaType.SYNCHRONOUS));
```

Figura 4.16: Critério de busca por canais de comunicação (tipo de canal)

Nestes exemplos, são criados dois tipos de critérios de busca. O primeiro indica que as pessoas que retornarem na busca devem possuir ao menos um meio do comunicação que seja do tipo síncrono (*MediaType.SYNCHRONOUS*). O segundo informa que a pessoa deve possuir, especificamente, o canal de comunicação cujo nome é ICQ.

A forma como estes critérios são criados durante a execução da aplicação que utiliza o MaPS, como já falado anteriormente, depende da implementação do framework. Assim, estes podem ser criados não explicitamente pelo usuário, mas por exemplo, através de um agente que o acompanha. O framework prevê e dá suporte a estas duas abordagens.

4.5 Considerações sobre o Capítulo

Neste capítulo foi introduzido o framework proposto neste trabalho, o framework MaPS. Foram apresentadas suas características, arquitetura, funcionalidades e requisitos. Devido ao seu escopo, o MaPS necessita de serviços do ambiente no qual está sendo executado, para tanto, também foram elencados pré-requisitos que o ambiente deve pro-

```
1
2 Criterion c = CriteriaBuilder.createContains(ProfileAttr.MEDIA, new
      Media("ICQ", null));
```

Figura 4.17: Critério de busca por canais de comunicação (definição de canal)

ver.

O capítulo finalizou com a exibição do desenvolvimento do protótipo do framework, destacando seus principais elementos.

Para realizar a avaliação do framework criou-se um plano de testes dividido em duas etapas. Estes planos estão descritos em detalhes no próximo capítulo.

5 AVALIAÇÃO

Tendo em vista a proposta apresentada pelo MaPS, este capítulo apresenta a metodologia traçada para a realização da avaliação do framework e seus resultados.

5.1 Metodologia de Avaliação

O presente trabalho apresenta um framework que auxilia no desenvolvimento de aplicações colaborativas em ambientes ubíquos. Mais especificamente, como já abordado, o modelo incrementa o processo de busca por pessoas e a forma como pode se dar a interação entre os participantes nestes ambientes.

MaPS é um framework, portanto uma proposta de infra-estrutura. Segundo Edwards et al. (2003) a avaliação de uma infra-estrutura é considerada problemática, uma vez que a mesma não é visível para o usuário final. Os autores colocam em seu trabalho que somente é possível avaliar as funcionalidades de um framework construindo aplicações que as utilizem e então avaliá-las, obtendo-se assim, uma avaliação indireta do framework. Contudo, como observado por eles, ao construir uma aplicação deve-se ter em mente que os usuários não estarão julgando apenas as características que desejamos, mas o software como um todo.

Desta maneira, para a avaliação desta proposta, optou-se por realizar uma abordagem sob duas perspectivas: **desenvolvimento de software** e **funcionalidade**. Nestas duas abordagens, preza-se pela avaliação de características específicas do framework proposto.

5.1.1 Perspectiva de Desenvolvimento de Software

A perspectiva de desenvolvimento de software tem como objetivo avaliar o MaPS em três aspectos:

- O framework suporta a construção de software com diferentes arquiteturas?

  MaPS não impõe uma arquitetura rígida para uma aplicação. Embora apresente uma estrutura definida, esta pode ser adaptada para diferentes organizações de aplicações.

- O framework suporta a utilização de diferentes perfis de usuários e mecanismos de canais de interação?

  Um dos requisitos funcionais do MaPS é a extensibilidade, ou seja, a possibilidade de acrescentar novos itens. O framework não possui suporte a um único modelo de perfil (embora, ressalte-se a disponibilização de informações básicas, como descrito na seção 4.1), assim, é possível utilizar o MaPS em qualquer aplicação que utilize perfis de usuários em sua execução, ou que disponibilize este tipo de informação.

- O framework é customizável?

  Este ponto tem como objetivo avaliar o esforço de programação necessário para estender o MaPS, considerando novos casos de utilização. Esta avaliação é dada em função da observação durante o desenvolvimento de aplicações que utilizem o MaPS.

Sob esta perspectiva, optou-se por desenvolver protótipos de duas aplicações semelhantes, que utilizam o MaPS. Através de diferenças de arquitetura e modelo de perfil de usuário entre elas os itens acima serão avaliados. Nas seções 5.2.1 e 5.2.2 estão descritas as aplicações que foram prototipadas para a avaliação do MaPS.

5.1.2 Perspectiva de Funcionalidade

A segunda abordagem, a perspectiva de funcionalidade, compreende as avaliações quanto às funcionalidades que o framework se propõe a suportar em uma aplicação colaborativa: a busca por pessoas e formas de comunicação entre estas. Para tanto são realizados testes com usuários, onde eles executam atividades que contemplem a proposta do MaPS.

Primeiramente, é apresentado aos usuários um cenário motivacional. Este descreve brevemente uma possível situação de utilização de uma aplicação que use as funcionalidades providas pelo framework. Com isso, os participantes podem compreender os objetivos do MaPS. O cenário apresentado é o seguinte:

> *"Joana é uma aluna do curso de Engenharia de Software. Atualmente ela está estudando, nas disciplinas em que cursa, Paradigmas de Programação.*
>
> *Esta semana, seu professor passou-lhes um trabalho para ser entregue sobre o paradigma Orientado a Objetos. Por isso, hoje Joana foi para Universidade pesquisar materiais e iniciar o seu trabalho.*
>
> *Então, Joana dirige-se para o laboratório a fim de utilizar os recursos disponíveis lá: estações para trabalho e Internet. Em todas as estações de trabalho do laboratório existe uma aplicação de IM (Instant Messaging) que possui integrada uma ferramenta de busca por pessoas. Ao acessar a es-*

tação, através de seu usuário, esta aplicação a identifica também e pode deduzir a sua localização no prédio.

Durante a execução do seu trabalho, Joana sente dificuldade em entender o conceito de polimorfismo, por isso decide buscar ajuda entre seus colegas e professores da Universidade. Utilizando a aplicação de pesquisa por pessoas, Joana indica que deseja encontrar pessoas que tenham conhecimentos em 'polimorfismo' para que possa interagir. Assim, o sistema, que é ciente do contexto, realiza a busca por pessoas que possuam cadastrado no seu perfil um conhecimento no tópico 'polimorfismo' e, adicionalmente, prioriza as pessoas que estão disponíveis neste momento para interagir com ela. Isto significa priorizar na lista de resultados, as pessoas que estejam fisicamente próximas ou que estejam disponíveis para uma interação online*. Contudo, não foi retornado nenhum resultado.*

*Nesta aplicação, reside também um agente, que permanece em background, e que au*xili*a seus usuários em suas pesquisas. O agente verifica que não foi retornado nenhum resultado, altera o filtro da busca e submete novamente a requisição. Ao invés de procurar por pessoas que possuam um conhecimento específico por polimorfismo, a busca é feita por termos semelhantes também. Assim, no novo resultado, aparecem pessoas que possuam conhecimentos em 'Orientação a Objetos', que é um termo que abrange 'polimorfismo'.*

Quando Joana recebe na sua tela o resultado da busca, vê uma mensagem, indicando que a busca foi expandida para retornar não só mais resultados, mas também informações relevantes, considerando o seu contexto. Dentre as pessoas que retornaram, Joana vê que seu colega Luciano possui conhecimentos em Orientação a Objetos e está no mesmo prédio que ela. Verifica também que ele está online *e que o meio de comunicação que ambos têm disponível e em comum é o Instant Messaging. Assim, envia-lhe uma mensagem e os dois começam a conversar."*

Até este momento, os participantes não interagem com nenhuma aplicação. Seguida a apresentação do cenário acima, os usuários executam uma atividade onde têm um papel de filtro de uma busca. Eles mesmos selecionam pessoas, de um conjunto previamente fornecido, em função de critérios de pesquisa. As etapas seguidas para esta avaliação são as seguintes:

1. apresentação de uma lista de pessoas, com seus perfis, para um usuário;

2. com base no cenário apresentado, é exibido para o participante um conjunto de critérios que devem ser satisfeitos na pesquisa por pessoas;

3. é solicitado ao usuário que indique quais pessoas poderiam retornar na consulta e por quê.

Após esta participação do usuário, são comparados os resultados que ele indicou com os resultados que uma aplicação construída com o MaPS retorna, utilizando-se os mesmos critérios de busca. Se houverem resultados que a aplicação retorne e o usuário não tenha previamente selecionado, é questionado ao mesmo se ele concorda com a escolha do software.

De posse dos dados fornecidos pelos usuários, é elaborada uma lista unificada de sugestões dadas e esta será relacionada com o resultado obtido através do protótipo.

A fim de comparar os resultados obtidos com as respostas dos usuários e o resultado fornecido pela aplicação, optou-se por utilizar o coeficiente de correlação de Spearman (Triola, 1999), dada a natureza dos dados dos testes. O coeficiente r_sé calculado em função das duas listas de resultados pela seguinte expressão:

$$r_s = 1 - \frac{6\sum_{i=1}^{n} d^2}{n(n^2-1)} \tag{5.1}$$

onde:

- d: diferença entre postos
- n: número de pares de postos

O coeficiente gera um valor no intervalo de 0 a 1, sendo que 1 indica uma correlação muito forte.

Com este teste, espera-se verificar o quão próximo o resultado de uma busca feita pela aplicação está do esperado pelos usuários. Ou seja, será testada a capacidade de um software que utilize o MaPS retornar resultados satisfatórios.

É importante ressaltar que as aplicações utilizadas nos testes têm, unicamente, como objetivo a avaliação da funcionalidade provida pelo framework. Entende-se que aspectos como usabilidade, interação homem-computador ou ainda a eficiência dos canais de comunicação utilizados sejam importantes de se considerar na construção de aplicações, contudo, para a avaliação dos objetivos do framework estes não serão consideradas.

5.2 Protótipos de Aplicações

Para avaliação do MaPS foram desenvolvidas duas aplicações semelhantes. As duas possuem os mesmos requisitos e utilizam, igualmente, o MaPS. Contudo, cada uma foi implementada de uma forma diferenciada (distinguidas na arquitetura e tecnologia envolvida).

As aplicações prototipadas com o MaPS têm como objetivo simplesmente buscar pessoas em um ambiente. Cada usuário acessa o programa através de um login e preenche um perfil que será utilizado para otimizar as buscas por pessoas. Ao acessar a aplicação, o usuário poderá realizar esta busca por pessoas em função de conhecimentos que elas possuem. A aplicação retorna uma lista de usuários que são compatíveis com os requisitos informados e ainda indica formas de comunicação que o usuário poderá usar para interagir com os outros participantes.

Na figura 5.1 são exibidos os requisitos do protótipo, mostrando o comportamento das entidades envolvidas.

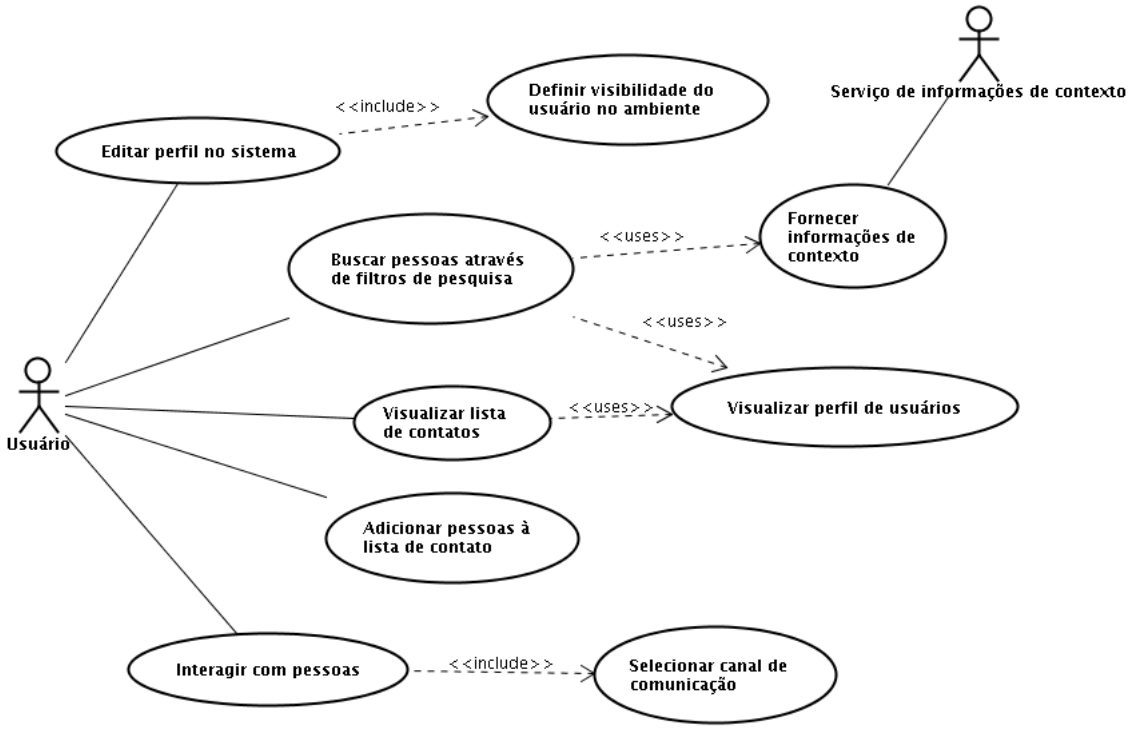


Figura 5.1: Casos de uso - Protótipo para avaliação

Como mostrado no diagrama, nesta aplicação, o objetivo do usuário é interagir com pessoas. Cada usuário pode se comunicar com qualquer pessoa que esteja no ambiente, esteja ela em sua lista de contatos ou não. Ao acessar a aplicação, o usuário define a sua visibilidade, indicando um dos estados abaixo:

- *Disponível*: usuário pode aparecer como resultado em buscas feitas por outras pessoas e também é visível para sua lista de contatos;
- *Privado*: usuário não aparece como resultado em buscas feitas por outras pessoas. Neste estado ele é visível somente para usuários de sua lista de contatos;
- *Invisível*: usuário não é visível para nenhuma pessoa do ambiente, contudo ele pode visualizar o estado de outros usuários (que estejam configurados para tal).

Para encontrar pessoas no ambiente, o usuário define filtros de pesquisa. A busca feita pela aplicação leva em consideração informações dos perfis dos usuários e ainda informações de contexto providas por um serviço externo. Para a avaliação, os serviços externos foram simulados, gerando informações pré-definidas e orientadas aos casos de teste.

A interação entre usuários poderá se dar com diferentes mídias de comunicação (troca de mensagens, e-mail, telefone), contudo, nenhuma através da aplicação. Nas consultas realizadas, o programa apenas informa quais são os melhores canais de comunicação para serem utilizados.

5.2.1 Protótipo da Aplicação Looking4U

Looking4 é uma aplicação *desktop* que foi desenvolvida utilizando a tecnologia Java e a IDE Eclipse. Por se tratar de uma aplicação que possui comunicação com um servidor ainda foi utilizado o servidor de aplicação JBoss 4.2.3[1]. Para o teste desta aplicação foi utilizado um notebook Pentium M Centrino 1.86GHz, com 1GB de memória RAM, disco de 100GB, Linux Ubuntu 8.04 como Sistema Operacional, e suporte a rede estruturada e *wireless*.

A arquitetura do protótipo desenvolvido pode ser observada na figura 5.2.

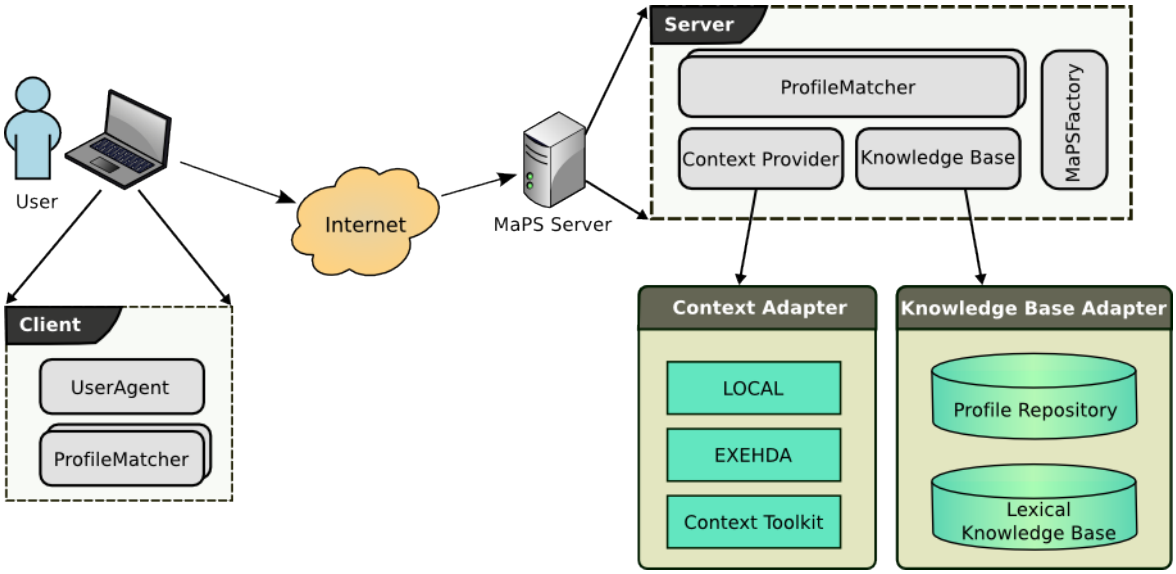


Figura 5.2: Arquitetura Looking4U

A organização apresentada contempla uma seleção multi-nível de *Profiles* (pipeline de instâncias de *ProfileMatcher*), como discutido anteriormente na seção 4.3.1, onde o pipeline é dividido entre cliente e servidor. Nesta organização, os clientes são os usuários portando diferentes tipos de dispositivos (por exemplo, *desktops*, laptops, PDAs), nos quais o framework está presente através de dois componentes: *UserAgent* e a instância

[1] http://www.jboss.com

```
1  # MaPS - Arquivo de configuracao
2  maps.debug=false
3
4  #Factory da instancia do MaPS
5  maps.factory=maps.factory.RMIServerFactory
6
7  ## Opcoes para ProfileMatcher
8  #Chain
9  maps.profileMatcher.0=maps.matcher.LocationAwareMatcher
10 maps.profileMatcher.1=maps.matcher.SimpleMatcher
11
12 ## Opcoes para ContextAdapter
13 maps.contextAdapter=com.looking4u.ContextAdapter
14
15 ## Opcoes para KnowledgeBase
16 maps.knowledgeBaseAdapter=com.looking4u.KnowledgeBase
```

Figura 5.3: Arquivo de configuração MaPS - Looking4U, servidor

```
1  # MaPS - Arquivo de configuracao
2  maps.debug=false
3
4  #Factory da instancia do MaPS
5  maps.factory=maps.factory.RMICLientFactory
6
7  ## Opcoes para ProfileMatcher
8  #Chain
9  maps.profileMatcher.0=com.looking4u.PresenceMatcher
10 #URL do servidor para acesso do cliente ProfileMatcher
11 maps.remoteServer=127.0.0.1
12
13 ## Opcoes para ContextAdapter
14 maps.contextAdapter=com.looking4u.ContextAdapter
15
16 ## Opcoes para KnowledgeBase
17 maps.knowledgeBaseAdapter=com.looking4u.KnowledgeBase
```

Figura 5.4: Arquivo de configuração MaPS - Looking4U, cliente

local de *ProfileMatcher*.

O *UserAgent* representa a lógica específica da aplicação colaborativa construída, sendo responsável pelas interações com o usuário. A instância local de *ProfileMatcher*, por sua vez, implementa o primeiro nível do mecanismo de pipeline de filtros, sendo acionado pelo *UserAgent* em função das ações realizadas pelo usuário.

Este exemplo de arquitetura baseia-se em uma solução distribuída, onde tanto o cliente quanto o servidor têm um papel ativo na busca por perfis. Os arquivos de configuração utilizados para este protótipo, que refletem a sua arquitetura, podem ser visto na figura 5.3 e 5.4.

Como pode ser observado nos arquivos de configuração, as classes *factory* utilizadas indicam o tipo de comunicação que será utilizado entre a aplicação cliente e servidor:

RMI. Estas classes são uma extensão das classes abstratas *ClientFactory* e *ServerFactory*, que por sua vez estendem *MaPSFactory* (classe responsável por criar as instâncias dos principais elementos utilizados pelo framework, como descrito na seção 4.4).

Como representado na figura 5.2, *ContextProvider* e *KnowledgeBase* são adaptadores que acessam serviços externos ao framework. Estes serviços são determinados pelo ambiente onde o framework está sendo utilizado. Na organização proposta na figura, *ContextProvider* abstrai o acesso aos *middleware* de contexto, exemplificados pelos *middleware* LOCAL (Barbosa et al., 2006, 2008), EXEHDA (Yamin, 2004) e ContextToolkit (Salber et al., 1999). Enquanto, *KnowledgeBase* abstrai o acesso a dois tipos de serviços: o repositório de perfis e a base de dados lexicais. Como exemplo de uma base de dados lexicais, pode-se citar o projeto WordNet (Miller, 1995).

Para fins de testes, os serviços externos foram simulados através de classes Java, dentro da aplicação. Como o acesso a estes serviços está encapsulado pelo framework, não há impacto para a aplicação o uso de um simulador de dados ou de um serviço real (falando-se sobre uma perspectiva de desenvolvimento de software).

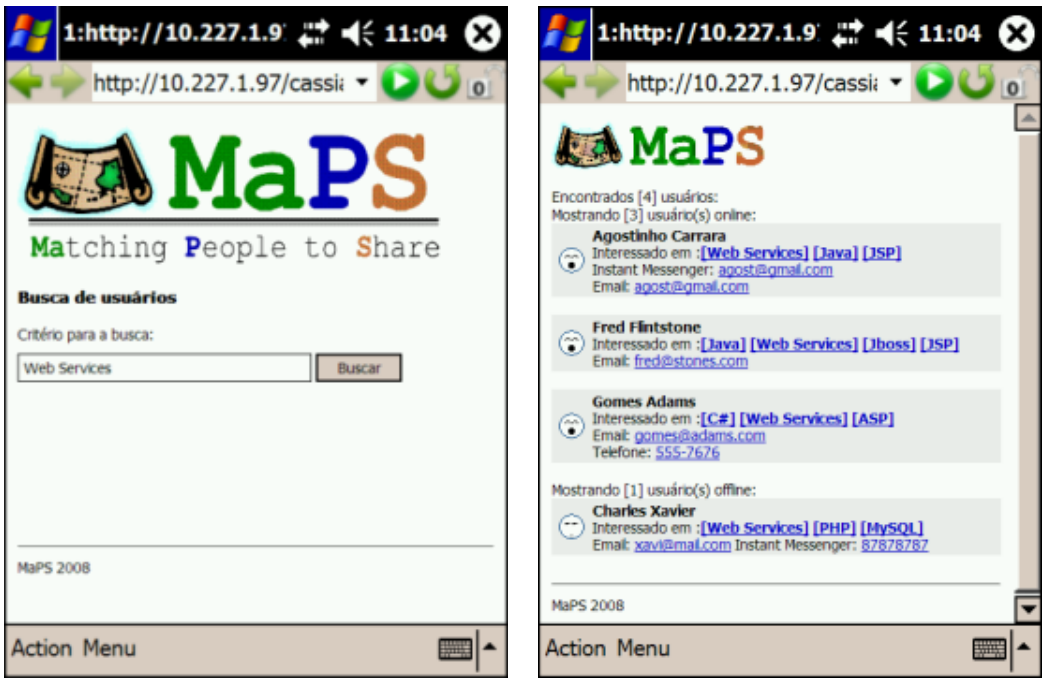
Figura 5.5: Tela principal - Looking4U

5.2.2 Protótipo de Aplicação PeopleFinder

PeopleFinder é uma aplicação *web*, desenvolvida com PHP, que acessa um servidor através de um *web service*. O servidor contém uma aplicação Java, que faz uso do MaPS para oferecer um serviço de busca de pessoas.

O desenvolvimento desta aplicação envolveu a utilização do servidor de aplicação JBoss 4.2.3 para a aplicação Java; e para a aplicação PHP o servidor PHP 5.2 e Apache

HTTP Server 2.2. Para o teste desta aplicação foi utilizado um notebook Pentium M Centrino 1.86GHz, com 1GB de memória RAM, disco de 100GB, Linux Ubuntu 8.04 como Sistema Operacional, e suporte a rede estruturada e *wireless* como servidor; e ainda um HP iPAQ hx4700 Pocket PC para acesso à aplicação via *browser*.



(a) Tela principal (b) Resultado de uma busca

Figura 5.6: Interface da aplicação PeopleFinder

Utilizando o aplicativo, o usuário tem a possibilidade de procurar pessoas que estejam cadastradas no sistema, através de uma busca por conhecimentos que eles possuem.Para cada resultado retornado, é informada uma lista de conhecimentos e que tipos de canais de comunicação o usuário pode utilizar para entrar em contato com a pessoa (figura 5.6).

MaPS foi usado nesta aplicação no modo local, ou seja, há um pipeline de filtros, mas ele está contido totalmente em um lugar só (servidor). O arquivo de configuração do framework para este caso pode ser observado na figura 5.7.

Ao contrário da aplicação anterior (seção 5.2.1), PeopleFinder utiliza o *factory* default do framework, que somente monta a cadeia de pipeline de instâncias de *ProfileMatcher*, não fazendo uso de um mecanismo especial de comunicação.

5.3 Resultados

Esta seção apresenta os resultados obtidos com os testes realizados seguindo-se a metodologia apresentada no início do capítulo.

```
1  # MaPS - Arquivo de configuracao
2  maps.debug=false
3
4  #Factory da instancia do MaPS
5  maps.factory=maps.factory.DefaultFactory
6
7  ## Opcoes para ProfileMatcher
8  #Chain
9  maps.profileMatcher.0=maps.matcher.LocationAwareMatcher
10 maps.profileMatcher.1=maps.matcher.SimpleMatcher
11
12 ## Opcoes para ContextAdapter
13 maps.contextAdapter=com.pplFinder.ContextAdapter
14
15 ## Opcoes para KnowledgeBase
16 maps.knowledgeBaseAdapter=com.pplFinder.KnowledgeBase
```

Figura 5.7: Arquivo de configuração MaPS - PeopleFinder

5.3.1 Aspectos de Desenvolvimento de Software

A execução do desenvolvimento dos dois protótipos de aplicações (Looking4U e PeopleFinder) teve como objetivo analisar alguns itens da proposta de avaliação de aspectos de desenvolvimento de software (seção 5.1.1). Os resultados obtidos desta avaliação são empíricos, originários a partir da experiência durante o processo de desenvolvimento dos protótipos.

Cada um dos protótipos foi desenvolvido utilizando uma arquitetura diferente. Looking4U, por ser uma aplicação *desktop*, possui a idéia de comunicação direta com um servidor. Sendo assim, para esta aplicação, MaPS foi utilizado no seu modo distribuído, fracionando o pipeline de filtros entre o cliente e o servidor. Para PeopleFinder a abordagem foi diferente. A aplicação acessa um *web service*, e por se tratar de um aplicativo *web* optou-se por manter toda a funcionalidade de seleção de usuários centralizada no servidor. Tanto o primeiro protótipo, quanto o segundo, fizeram uso do MaPS e suas diferentes organizações não prejudicaram a funcionalidade provida pelo framework.

Por ter um aspecto de comunicação remota, Looking4U utilizou RMI e para tanto utilizou classes *factory* (instanciadas do próprio MaPS) que implementam esta comunicação. Já PeopleFinder trabalha com *web services*, então possui uma preocupação em disponibilizar uma interface externa, que será acessada pelos seus clientes. A diferença nos dois casos é que MaPS não oferece um suporte à construção de um *web service*, assim, este trabalho ficou a cargo do utilizador do framework. Entretanto, utilizando ou não mecanismos de comunicação disponível pelo MaPS, nota-se que o processo de busca não é comprometido, podendo obter resultados semelhantes em ambas aplicações.

Através da análise feita durante o desenvolvimento destes protótipos para o estudo de caso, percebe-se que o framework atende a diferentes requisitos, oferecendo mecanismos para sua customização e expansão.

5.3.2 Aspectos de Funcionalidade

Para a avaliação de funcionalidade do framework, foram realizados testes com uma população de 12 indivíduos.

A fase de coleta de dados com os participantes compreendeu três etapas:

- Apresentação de um cenário motivacional, relatando a necessidade de uma aplicação de busca por pessoas;
- Apresentação de um conjunto de personagens de um ambiente, indicando-se que um deles deseja realizar uma pesquisa por outros personagens que possuam um determinado assunto;
- Participantes selecionam quatro personagens, em ordem decrescente de importância, que julgam que possuam condições de colaborar com o personagem que realiza a busca.

Com o intuito de não tornar os testes cansativos para os participantes, mas ao mesmo tempo, apresentar dados relevantes, foi definido um conjunto de 16 personagens no ambiente simulado. Cada um deles possui um perfil, que pôde ser consultado pelos participantes.

Os perfis dos personagens possuem informações como seu estado no momento (*online* ou *offline*), os conhecimentos que estes possuem e que tipos de mídias de comunicação têm disponível. A figura 5.8 mostra o modelo de perfil apresentado para os participantes.

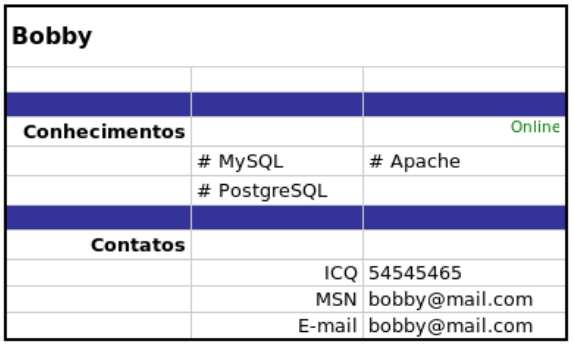


Figura 5.8: Modelo de perfil dos personagens

Ao final dos experimentos, foi calculado um nível de aptidão para cada personagem do ambiente simulado, refletindo a preferência dos indivíduos que participaram do experimento. Para tal, foi desenvolvida e adotada a fórmula

$$Apt(i) = \sum_{n=1}^{x} (F_n(i) * 2^{x-n}) \tag{5.2}$$

onde, o nível de aptidão do personagem i pertencente ao ambiente simulado $(Apt(i))$ é calculado pelo somatório ponderado da freqüência com que este indivíduo aparece em

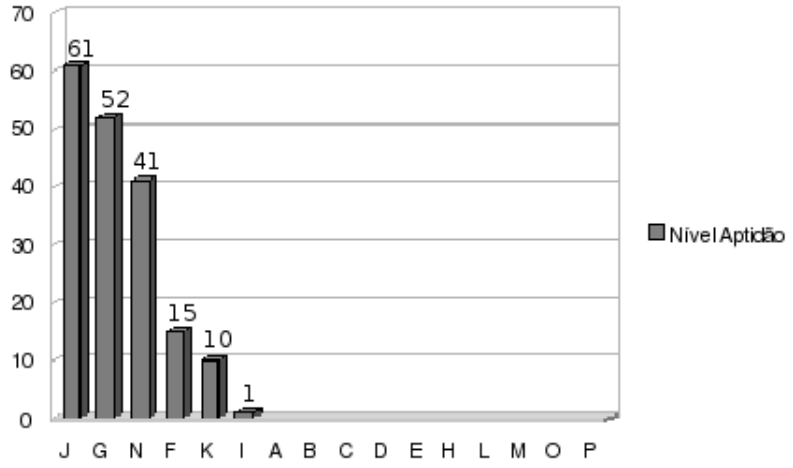


Figura 5.9: Aptidões dos personagens

cada uma das n posições de preferência ($F_n(i)$). O peso atribuído a cada elemento do somatório (2^{x-n}) segue uma progressão geométrica, visando destacar os elementos mais aptos dos demais. Como já citado anteriormente, os voluntários selecionaram 4 personagens, portanto x, na fórmula acima, possui este valor.

Por exemplo, considerando as respostas de todos os voluntários, o personagem G, do ambiente simulado, obteve a quantidade de votos para cada um das posições apresentada na tabela 5.1.

Tabela 5.1: Quantidade de votos para um personagem dada pelos voluntários

	Quantidade de votos			
Personagem	**Primeira Posição**	**Segunda posição**	**Terceira posição**	**Quarta posição**
G	3	5	4	0

A tabela indica que 3 pessoas tiveram o personagem G na primeira posição de suas listas, 5 diferentes pessoas o colocaram como segunda escolha, 4 o selecionaram como terceira opção e nenhuma o selecionou na quarta posição. Pode-se notar que este personagem participou da lista de 12 pessoas diferentes, ou seja, todos os voluntários o selecionaram em alguma posição.

Aplicando-se a fórmula de aptidões aos personagens, em função dos resultados fornecidos pelos voluntários, obteve-se os valores apresentados no gráfico da figura 5.9.

Dos 16 personagens do ambiente simulado, 6 apareceram ao menos uma vez em alguma lista fornecida pelos voluntários. O personagem J foi o que apareceu mais vezes e mais próximo das primeiras posições das listas, obtendo o maior valor de aptidão.

Com base nos valores de aptidão calculados para cada personagem, gerou-se uma lista ordenada resultante das respostas dos voluntários. Os quatro personagens com maior nível de aptidão formam esta nova lista (em ordem de decrescente importância: personagem J,

G, N e F).

Em paralelo a esta atividade, foi executada a mesma ação de busca realizada pelos participantes, mas utilizando-se um protótipo desenvolvido. A aplicação utilizou o mesmo conjunto de personagens e o mesmo critério de pesquisa que os apresentados para os usuários dos testes. Como resultado desta pesquisa, obteve-se a seguinte lista (em ordem de decrescente importância): J, G, N e F, que representa a mesma lista resultante das respostas dadas pelo usuário. Assim, torna-se desnecessário a aplicação do coeficiente de correlação, já que as duas listas são semelhantes. Com isso, verifica-se que o protótipo retornou indicações condizentes e relevantes para o problema proposto.

Confrontando os dados provenientes dos voluntários com o resultado do protótipo, pôde-se obter as seguintes estatísticas:

- 1 (8.33%) pessoa indicou exatamente o mesmo resultado que o protótipo apresentou;
- 5 (41.67%) pessoas indicaram os mesmos dois primeiros resultados que o protótipo apresentou (não na mesma ordem);
- 7 (58.33%) pessoas indicaram os mesmos três primeiros resultados que o protótipo apresentou (não na mesma ordem);
- todos apresentaram em sua seleção (em alguma posição) o primeiro e o segundo selecionados pelo protótipo;
- 11 (91.67%) pessoas apresentaram em sua seleção (em alguma posição) o primeiro, segundo e terceiro selecionados pelo protótipo;
- 7 (58.33%) pessoas apresentaram em sua seleção (em alguma posição) os mesmos resultados que o protótipo.

Com estas métricas, pode-se notar que dentre os participantes do experimento, todos apresentaram em suas listas, em qualquer ordem, as duas primeiras indicações feitas pelo protótipo construído com o auxílio do MaPS. E somente um participante não apresentou em sua lista, em qualquer ordem, os três primeiros resultados selecionados pelo protótipo. Isto significa que para mais de 90% da população do experimento o MaPS acertou em suas indicações, considerando-se seus três primeiros resultados, de quatro fornecidos.

5.4 Considerações sobre o Capítulo

Este capítulo descreveu de forma objetiva a metodologia de avaliação do framework seguida. Foram apresentadas as duas abordagens que foram utilizadas (desenvolvimento de software e funcionalidade) e quais pontos estarão sendo mensurados nos testes. Também foram apresentados os dois protótipos de aplicações desenvolvidos com o auxílio

do MaPS, apresentando-se sua modelagem e implementação. E ainda os resultados das avaliações obtidos.

O próximo capítulo tem como objetivo dar um fechamento ao trabalho aqui apresentado, destacando suas principais contribuições, limitações, bem como sugestões de trabalhos futuros.

6 CONSIDERAÇÕES FINAIS

Este capítulo apresenta uma reflexão sobre as principais conclusões e contribuições proporcionadas pelo MaPS. Complementando tal discussão ainda são exibidos trabalhos futuros.

6.1 Contribuições

Atualmente, existem soluções voltadas para o suporte ao desenvolvimento de aplicações colaborativas. Estas soluções vão desde frameworks conceituais (Zhang and Jin, 2005; Zhang et al., 2005a) até plataformas de serviços (Bergenti et al., 2002; Divitini et al., 2004; Farshchian and Divitini, 2005) que têm como intuito auxiliar no desenvolvimento de aplicações deste tipo. Entretanto, como caracterizado no estudo realizado sobre tais soluções (capítulo 3), os processos envolvidos nos passos elementares de uma colaboração, seleção de pares e canais de comunicação, não foram completamente explorados. De forma que existe margem para uma contribuição científica neste assunto.

MaPS vem preencher esta lacuna, oferecendo um suporte especializado a estes aspectos na criação de software colaborativo através do framework MaPS. Na tabela 6.1 é apresentada novamente a tabela comparativa do projetos estudados, mas confrontando-se com o MaPS.

MaPS foi criado para oferecer um novo modelo de busca de pessoas, em um ambiente ubíquo, com o objetivo de colaborar. E para isso, faz uso de informações de contexto e perfis de usuários, a fim de tornar este processo mais eficiente. Outros projetos também utilizam dados de perfis de usuários, no entanto, nem todos usam estas informações para a busca de usuários.

uLearning é um projeto que propõe a busca por pessoas, como MaPS, e ressalta a sua importância no processo de colaboração. Entretanto, não trata do problema de seleção de canais de comunicação. Este projeto apresenta um modelo conceitual e descreve uma aplicação pronta que implementa os conceitos propostos, não oferecendo recursos para desenvolvedores.

Outro projeto que apresenta a funcionalidade de busca por pares é o baseado em redes

Tabela 6.1: Tabela comparativa dos projetos estudados e do framework MaPS

Característica	uLearning	UbiCollab	P2P e Redes Sociais	Collaborator	MaPS
Utiliza um modelo de dados do usuário	x	x	x	x	x
Considera informações de contexto	x	x		x	x
Permite a utilização de diversos tipos de canais de comunicação	x	x		x	x
Oferece um mecanismo de busca por pessoas	o	o	x		x
Leva em consideração os canais de comunicação na busca por pessoas	x				x
Oferece uma biblioteca ou serviços destinados ao desenvolvimento de aplicações		x		x	x
Pode ser estendido para outros domínios ou tipos de aplicação	x	x	x	x	x

sociais. Neste trabalho, o conceito de busca por recursos (sejam arquivos ou pessoas) está presente e é dada uma grande importância a ele. Todavia, mesmo sendo uma proposta que está inserida em um ambiente ubíquo, esta ação de pesquisa não utiliza informações de contexto, ou seja, não faz uso de um recurso que pode estar disponível pelo ambiente e que pode incrementar seus resultados.

UbiCollab e Collaborator são projetos que, assim como MaPS, se propõem a fornecer subsídios aos desenvolvedores para construírem seus próprios sistemas, aplicando os conceitos apresentados. Contudo, apesar de oferecer uma base para tal, estes ambientes não descrevem explicitamente um processo de busca por colaboradores. Em um ambiente onde seus usuários não têm conhecimento de todos os participantes envolvidos, esta é uma das funcionalidades mais básicas para se prover um cenário propício para colaboração.

O diferencial do MaPS em relação a outros projetos que tratam do suporte à funcionalidade de seleção de pessoas para uma colaboração está em considerar informações distintas (não utilizando só dados que o usuário informa explicitamente), que melhoram a qualidade da busca, retornando potenciais colaboradores que são pertinentes, considerando o contexto em que a busca é realizada. Dados como tipos de canais de comunicação disponíveis, perfis de usuários e localização dos mesmos não são consideradas em sua totalidade nos trabalhos relacionados, tratando-se de um aspecto que distingue MaPS dos demais trabalhos, caracterizando-o por sua principal contribuição.

6.2 Conclusões

Computadores são utilizados como suporte à colaboração há décadas, mas este uso vem sendo adaptado ao longo deste tempo em função das tecnologias disponíveis. Nos últimos anos assistimos o desenvolvimento do paradigma da computação ubíqua e como ele pode modificar a forma como tratamos a colaboração. Um ambiente ubíquo pode

prover recursos antes não existentes para a elaboração de sistemas colaborativos.

Mobilidade, poder computacional em qualquer lugar e a qualquer hora, ciência do contexto, estas são apenas algumas características que a computação ubíqua disponibiliza para a promoção da colaboração. Contudo, o bom uso destes aspectos ainda é um campo de pesquisa vasto na computação.

Neste cenário, MaPS foi pensado para melhor explorar as características providas em um ambiente ubíquo, especificamente, nos passos elementares do processo de colaboração. E para que possa ser possível utilizar este trabalho no desenvolvimento de aplicações, optou-se por concebê-lo através de uma das técnicas mais utilizadas de reuso de software: um framework.

6.3 Trabalhos Futuros

Um framework não está finalizado em suas primeiras versões. Segundo Pree (2000), um framework precisa ser especializado diversas vezes, continuamente, com o objetivo de detectar suas partes falhas e incrementá-las, e ainda identificar novos *hot-spots*, que em um primeiro momento não foram descobertos. Nesse sentido, é interessante que novas aplicações, de diferentes domínios, sejam desenvolvidas fazendo uso do MaPS com o intuito de aprimorar o framework.

Com a possibilidade de se utilizar informações de contexto nas suas pesquisa, MaPS abre espaço para diversos trabalhos futuros. Diferentes protótipos que utilizem estes dados (como localização, por exemplo) podem ser explorados. Como trabalho futuro destaca-se ainda uma extensão do MaPS para a utilização de sistemas de recomendação, onde os usuários avaliariam a atuação de seus colaboradores e suas proficiências. Utilizando esta informação como filtro, os processos de busca podem ser otimizados.

Outro trabalho futuro que visa incrementar a busca por colaboradores é uma extensão do MaPS onde acesse informações de histórico de colaborações. Estas interações já realizadas podem servir como informações preferências na momento de se indicar colaboradores para um usuário.

Por fim, são válidas também as extensões do framework para o suporte nativo a mais de um tipo de comunicação cliente-servidor. Hoje, MaPS possui um pacote utilitário para acesso via RMI, contudo poderiam ainda ser acrescentados mecanismos para acesso, por exemplo, via sockets.

REFERÊNCIAS

Bannon, L. J. and Schmidt, K. (1989). Cscw: Four characters in search of a context. pages 358–372, Gatwick, London, UK. Computer Sciences House, Slough, UK.

Barbosa, D. N. F. (2007). *Um Modelo de Educação Ubíqua Orientado à Consciência do Contexto do Aprendiz*. Doutorado em ciência da computação, Instituto de Informática, PPGC/UFRGS, Porto Alegre.

Barbosa, J., Hahn, R., Rabello, S., and Barbosa, D. (2008). Local: a model geared towards ubiquitous learning. pages 432–436, Portland, OR, USA. ACM.

Barbosa, J. V., Hahn, R., Rabello, S., and Barbosa, D. N. F. (2006). Local: Um modelo para suporte a aprendizagem consciente de contexto. In *Simpósio Brasileiro de Informática na Educação*, pages 437–446, Brasília. Porto Alegre: SBC.

Bergenti, F., Costico, S., and Poggi, A. (2003). A portal for ubiquitous collaboration. In *Conference on Advanced Information Systems Engineering*, Klagenfurt, Austria.

Bergenti, F., Poggi, A., and Somacher, M. (2002). A collaborative platform for fixed and mobile networks. *Communications of the ACM*, 45:39–44.

Brok, J., Kumar, B., Meeuwissen, E., and Batteram, H. J. (2006). Enabling new services by exploiting presence and context information in ims. *Bell Labs Technical Journal*, 10:83–100.

Campos, F. C. A., Santoro, F. M., Borges, M. R. S., and Santos, N. (2003). *Cooperação e aprendizagem on-line*. DP&A.

Carroll, J. M., Neale, D. C., Isenhour, P. L., Rosson, M. B., and McCrickard, D. S. (2003). Notification and awareness: synchronizing task-oriented collaborative activity. *Int. J. Hum.-Comput. Stud.*, 58:605–632.

Chen, I. and Yang, S. (2006). Peer-to-peer knowledge sharing in collaboration supported virtuallearning communities. In *Advanced Learning Technologies, 2006. Sixth International Conference on*, pages 807–809.

Crespo, S. (2000). *Composição em WebFrameworks*. Phd thesis in computer science, PUC-Rio.

Dey, A. K. and Abowd, G. D. (2000). Towards a better understanding of context and context-awareness. In *Conference on Human Factors in Computing Systems*, The Hague, The Netherlands.

Diamadis, E. T. and Polyzos, G. C. (2004). Efficient cooperative searching on the web: system design and evaluation. *International Journal of Human-Computer Studies*, 61:699–724.

Divitini, M., Farshchian, B. A., and Samset, H. (2004). Ubicollab: collaboration support for mobile users. pages 1191–1195, Nicosia, Cyprus. ACM.

Edwards, W. K., Bellotti, V., Dey, A. K., and Newman, M. W. (2003). The challenges of user-centered design and evaluation for infrastructure. pages 297–304, Ft. Lauderdale, Florida, USA. ACM.

Elliott, M. A. (2007). *Stigmergic Collaboration - A Theoretical Framework for Mass Collaboration*. PhD thesis, Victorian College of the Arts - The University of Melbourne.

Ellis, C. A., Gibbs, S. J., and Rein, G. (1991). Groupware: some issues and experiences. *Communications of the ACM*, 34:39–58.

Farshchian, B. and Divitini, M. (2005). Ubicollab: improving collaboration with location services. In *Pervasive Services, 2005. ICPS '05. Proceedings. International Conference on*, pages 417–420.

Farshchian, B. A. and Divitini, M. (2009). *Collaboration Support for Mobile Users in Ubiquitous Environments*. Springer, Heidelberg.

Fayad, M. E., Schmidt, D. C., and Johnson, R. E. (1999). *Building application frameworks : object-oriented foundations of framework design*. New York: John Wiley & Sons.

Fontoura, M., Pree, W., and Rumpe, B. (2000). *UML-F: A Modeling Language for Object-Oriented Frameworks*, pages 63–82.

Fontoura, M. F. M. C. (1999). *A Systematic Approach for Framework Development*. Phd thesis in computer science, PUC-Rio.

Fuks, H., Raposo, A., Gerosa, M. A., Pimental, M., and Lucena, C. J. P. (2007). *The 3C Collaboration Model*, page 750. Information Science Reference, Texas A&M International University, USA, ned kock edition.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (2000). *Padrão de Projetos - Soluções reutilizáveis de software orientado a objetos*. Bookman, Porto Alegre.

Gerosa, M. A., Fuks, H., and Lucena, C. J. P. (2003). Suporte à percepção em ambientes digitais de aprendizagem. *Revista Brasileira de Informática na Educação*, 11.

Grudin, J. (1994). Computer-supported cooperative work: history and focus. *Computer*, 27:19–26.

Hightower, J. and Borriello, G. (2001). Location systems for ubiquitous computing. *Computer*, 34:57–66.

Izadi, S., Coutinho, P., Rodden, T., and Smith, G. (2002). The fuse platform: Supporting ubiquitous collaboration within diverse mobile environments. *Automated Software Engineering*, 9:167–186.

Johansen, R. (1988). *Groupware: Computer Support for Business Teams*. The Free Press, New York, NY, USA.

Johnson, R. E. (1997). Components, frameworks, patterns. pages 10–17, Boston, Massachusetts, United States. ACM.

Mattsson, M. (1996). Object-oriented frameworks - a survey of methodological issues.

Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, 38:39–41.

Naismith, L. and Smith, P. (2004). Context-sensitive information delivery to visitors in a botanic garden. In *EDMEDIA World Conference on Educational Multimedia, Hypermedia and Telecommunications*, volume 2004, pages 5525–5530, Lugano, Switzerland.

Ooi, V. B. Y. (1998). *Computer Corpus Lexicography*. Edinburgh University Press, Edinburgh.

Park, W.-I., Kang, S.-J., Lee, Y.-D., Choi, H.-S., Kang, S.-H., Choi, M., Sohn, K., Kim, Y.-K., and Kang, J.-H. (2007). A context-based collaboration system in ubiquitous environments. In *Convergence Information Technology, 2007. International Conference on*, pages 101–107.

Penichet, V., Marin, I., Gallud, J., Lozano, M., and Tesoriero, R. (2007). A classification method for cscw systems. *Electronic Notes in Theoretical Computer Science*, 168:237–247.

Pree, W. (1994). *Design Patterns for Object-Oriented Software Development*. Addison Wesley Longman, 1st edition.

Pree, W. (2000). Hot-spot-driven framework development. In Fayad, M., Schmidt, D., and Johnson, R., editors, *Building Application Frameworks: Object-Oriented Foundations of Framework Design*. Wiley & Sons, New York City.

Resta, P. E. (1995). Project circle: student mentors as a strategy for training and supporting teachers in the use of computer-based tools for collaborative learning. pages 280–282, Indiana Univ., Bloomington, Indiana, United States. Lawrence Erlbaum Associates, Inc.

Saha, D. and Mukherjee, A. (2003). Pervasive computing: A paradigm for the 21st century. *Computer*, 36:25–31.

Salber, D., Dey, A. K., and Abowd, G. D. (1999). The context toolkit: aiding the development of context-enabled applications. pages 434–441, Pittsburgh, Pennsylvania, United States. ACM.

Satyanarayanan, M. (1996). Fundamental challenges in mobile computing. pages 1–7, Philadelphia, Pennsylvania, United States. ACM.

Satyanarayanan, M. (2001). Pervasive computing: vision and challenges. *Personal Communications, IEEE [see also IEEE Wireless Communications]*, 8:10–17.

Schilit, B., Adams, N., and Want, R. (1994). Contextaware computing applications. Santa Crus, CA, US.

Schmidt, D., Stal, M., Rohnert, H., and Buschmann, F. (2001). *Pattern-oriented software architecture : patterns for concurrent and networked objects*. Chichester: John Wiley & Sons, 1st edition.

Shafer, S. A. N., Brumitt, B., and Cadiz, J. (2001). Interaction issues in context-aware intelligent environments. *Human-Computer Interaction*, 16:363 – 378.

Stahl, G. (2002). Groupware goes to school. In *8th International Workshop on Groupware: Design, Implementation, and Use, CRIWG 2002*, volume 2440 of *Lecture Notes in Computer Science*, pages 7–24, La Serena, Chile. Berlin: Springer.

Syvanen, A., Beale, R., Sharples, M., Ahonen, M., and Lonsdale, P. (2005). Supporting pervasive learning environments: adaptability and context awareness in mobile learning. In *Wireless and Mobile Technologies in Education, 2005. WMTE 2005. IEEE International Workshop on*, page 3 pp.

Toivonen, S., Kolari, J., and Laakko, T. (2003). Facilitating mobile users with contextualized content. *Proceedings of the 1998 Winter Simulation Conference Proceedings*, pages 124–134.

Triola, M. F. (1999). *Introdução à Estatística*. Livros Técnicos e Científicos Editora S. A., Rio de Janeiro, 7 edition.

Weiser, M. (1991). The computer for the 21st century. *Scientific American*, 265(9).

Weiser, M. (1993). Some computer science issues in ubiquitous computing. *Communications of the ACM*, 36:75–84.

Yamin, A. C. (2004). *Arquitetura para um Ambiente de Grade Computacional direcionado às Aplicações Distribuídas, Móveis e Conscientes do Contexto da Computação Pervasiva*. Tese (doutorado em ciência da computação), Instituto de Informática, Universidade Federal do Rio Grande do Sul.

Yang, S. J. (2006). Context aware ubiquitous learning environments for peer-to-peer collaborative learning. 9:188–201.

Yang, S. J. and Chen, I. Y. (2008). A social network-based system for supporting interactive collaboration in knowledge sharing over peer-to-peer network. *International Journal of Human-Computer Studies*, 66:36–50.

Zhang, G. and Jin, Q. (2005). Research on collaborative service solution in ubiquitous learning environment. In *Parallel and Distributed Computing, Applications and Technologies, 2005. PDCAT 2005. Sixth International Conference on*, pages 804–806.

Zhang, G., Jin, Q., and Lin, M. (2005a). A framework of social interaction support for ubiquitous learning. In *Advanced Information Networking and Applications, 2005. AINA 2005. 19th International Conference on*, volume 2, pages 639–643 vol.2.

Zhang, G., Jin, Q., and Shih, T. (2005b). Peer-to-peer based social interaction tools in ubiquitous learning environment. In *Parallel and Distributed Systems, 2005. Proceedings. 11th International Conference on*, volume 1, pages 230–236 Vol. 1.