UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
COORDENAÇÃO DE INFORMÁTICA
CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS PARA INTERNET

JOSÉ TEODORO DA SILVA

UMA ARQUITETURA EXTENSÍVEL PARA LOCALIZAÇÃO DE ESPECIALISTAS EM CENÁRIOS DE DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE

TRABALHO DE CONCLUSÃO DE CURSO

CAMPO MOURÃO
2011

JOSÉ TEODORO DA SILVA

UMA ARQUITETURA EXTENSÍVEL PARA LOCALIZAÇÃO DE ESPECIALISTAS EM CENÁRIOS DE DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina Trabalho de Conclusão de Curso II, do Curso Superior de Tecnologia em Sistemas para Internet da Coordenação de Informática da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção de título de Tecnólogo.

Orientador: Prof. MSc. Igor F. Steinmacher

CAMPO MOURÃO
2011

ATA DA DEFESA DO TRABALHO DE CONCLUSÃO DE CURSO

As **vinte e uma horas** do dia **vinte e quatro de novembro de dois mil e onze** foi realizada no Miniauditório da UTFPR-CM a sessão pública da defesa do Trabalho de Conclusão do Curso Superior de Tecnologia em Sistemas para Internet do acadêmico **José Teodoro da Silva** com o título **UMA ESTRUTURA EXTENSÍVEL PARA LOCALIZAÇÃO DE ESPECIALISTAS EM CENÁRIOS DE DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE**. Estavam presentes, além do acadêmico, os membros da banca examinadora composta pelo professor **Me. Igor Fábio Steinmacher** (Orientador-Presidente), pelo professor **Me. Ivanilton Polato** e pelo professor **Me. Igor Scaliante Wiese**. Inicialmente, o aluno fez a apresentação do seu trabalho, sendo, em seguida, arguido pela banca examinadora. Após as arguições, sem a presença do acadêmico, a banca examinadora o considerou **Aprovado** na disciplina de Trabalho de Conclusão de Curso e atribuiu, em consenso, a nota ____ . Este resultado foi comunicado ao acadêmico e aos presentes na sessão pública. A banca examinadora também comunicou ao acadêmico que este resultado fica condicionado à entrega da versão final dentro dos padrões e da documentação exigida pela UTFPR ao professor Responsável do TCC no prazo de **quinze dias**. Em seguida foi encerrada a sessão e, para constar, foi lavrada a presente Ata que segue assinada pelos membros da banca examinadora, após lida e considerada conforme.

Observações:

_____

_____

_____

Campo Mourão, 24 de novembro de 2011.

Prof. Me. Ivanilton Polato Membro	Prof. Me. Igor Scaliante Wiese Membro	Prof. Me. Igor Fábio Steinmacher Orientador

A folha de aprovação assinada encontra-se na coordenação do curso.

RESUMO

SILVA, José Teodoro. Uma Arquitetura Extensível para Localização de Especialistas em Cenários de Desenvolvimento Distribuído de Software. 54 f. Trabalho de Conclusão de Curso – Curso Superior de Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná – UTFPR. Campo Mourão, 2011.

O desenvolvimento distribuído de software (DSD) permite que as organizações aproveitem capital humano distribuído por todo o mundo. Entretanto a comunicação, que é essencial para o desenvolvimento, é prejudicada pela imposição da distância geográfica. Para resolver esses problemas existem ferramentas que auxiliam na troca de informações eficientemente. Mas, estas ferramentas são inflexíveis e criadas para agir em domínios específicos. Neste trabalho foi desenvolvida uma arquitetura extensível que possibilita a identificação dos especialistas em determinados assuntos utilizando diferentes algoritmos e que possibilite a adição de novos métodos de recomendação. Além de permitir a extensibilidade a ferramenta abre espaço para extensões que solucionem os problemas gerados pelas diferenças pessoais, culturais e temporais existentes em ambientes de DSD.

Palavras-chaves: Recomendação de *Experts*. Desenvolvimento Distribuído de Softwares. Serviços *Web*. Engenharia de software.

ABSTRACT

SILVA, José Teodoro. An Extensible Architecture to find Experts in Distribuited Development Software Enviroments. 54 f. Trabalho de Conclusão de Curso – Curso Superior de Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná – UTFPR. Campo Mourão, 2011.

Distributed software development (DSD) enables organizations to leverage human capital distributed throughout the world. However the communication, that is essential to software development, is hindered by the geographical distance. To solve these problems there are some tools that aims to support information exchange efficiently. But, these tools are inflexible and are designed to attend a specific scenario. In this work we developed an extensible architeture that facilitates the identification of experts in certain subjects, using different algorithms and enabling the addition of new recommendation methods. Besides enabling different methodologies, the proposed tool is extensible enough to support different clients, addressing the issues caused by personal, cultural and temporal differences found in DSD environments.

Keywords: Recommendation. Distributed Software Development. Web services. Software Engineering.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

A indústria de desenvolvimento de software tem ganhado vantagens competitivas em termos de custo e qualidade por utilizar profissionais espalhados pelo mundo (ROBINSON; KALAKOTA, 2004). Estas vantagens aliadas aos avanços tecnológicos na área de telecomunicações fomentaram uma nova abordagem de desenvolvimento colaborativo: o desenvolvimento distribuído de software (DULLEMOND; VAN GAMEREN; VAN SOLINGEN, 2010). Com ele, organizações podem utilizar recursos humanos distribuídos por todo globo sem a preocupação com a distância geográfica (TRINDADE; MORAES; MEIRA, 2008) (EHRLICH; CHANG, 2006). A comunicação entre indivíduos dispersos geograficamente deixou de ser realizada pessoalmente e passou a ser dependente de ferramentas que promovem a comunicação (MOCKUS; HERBSLEB, 2002).

A comunicação é essencial para o desenvolvimento distribuído, uma vez que os desenvolvedores necessitam de informações precisas para realizar seu trabalho (TRINDADE; MORAES; MEIRA, 2008). Eles precisam estar cientes do andamento do projeto para reduzir os custos de desenvolvimento, evitar conflitos e retrabalho (KWAN; DAMIAN; STOREY, 2006). Todas as informações relativas ao desenvolvimento do projeto precisam ser compartilhadas entre os membros, e a dificuldade na transferência destes conhecimentos pode afetar o desempenho da equipe (EHRLICH; CHANG, 2006). O uso de diferentes meios de comunicação possibilita que os vários indivíduos que compõem uma equipe troquem informações necessárias para a realização de suas tarefas.

O compartilhamento eficiente de informações é um fator crítico durante o desenvolvimento de software (TRINDADE; MORAES; MEIRA, 2008). Este compartilhamento é afetado pela distância física e temporal no desenvolvimento distribuído, e a troca de informações de forma precária incrementa o tempo de desenvolvimento do projeto e dificulta a tarefa de localizar especialistas (*experts*) (MORAES *et al.*, 2010). Essa troca de informações não é eficiente quando um indivíduo não consegue identificar quem pode ajudá-lo na solução de um problema. Os desenvolvedores precisam de um meio para encontrar quem possa ajudá-los rapidamente (TRINDADE; MORAES; MEIRA, 2008). O tempo demandado para

executar esta busca pode ser reduzido com o uso de ferramentas que promovam a comunicação eficiente e a identificação de especialistas.

A necessidade de identificação dos especialistas foi verificada empiricamente. Uma pesquisa de campo foi conduzida por meio de um questionário direcionado a membros de projetos distribuídos de software. Os entrevistados qualificaram a importância de dez diferentes informações durante a execução de uma tarefa. Os resultados demonstram a importância da identificação dos especialistas durante o desenvolvimento. Além disso, foi verificado que os entrevistados realizam esta busca manualmente, em alguns casos efetuam a busca em mais de uma fonte de dados. O tempo gasto com esta busca poderia ser investido na execução das tarefas. O uso de ferramentas que auxiliem nessa busca pode diminuir este tempo gasto.

Existem várias ferramentas que têm por objetivo facilitar a busca por especialistas em projetos de software de acordo com determinados artefatos ou assuntos (MOCKUS; HERBSLEB, 2002), (SHAMI; EHRLICH; MILLEN, 2006), (BEGEL; DELINE; ZIMMERMANN, 2010), (TRINDADE; MORAES; BARBOSA, 2009). Entretanto, cada uma destas ferramentas possuem metodologias específicas para identificação dos especialistas e apenas um meio de exibição dos resultados. Elas não são suficientemente extensíveis e não acomodam novos requisitos ou mudanças na aplicação (JOHNSON *et al.*, 2005). Portanto seu uso é limitado a contextos e situações específicas.

O objetivo deste trabalho é apresentar uma arquitetura para identificação de especialistas em projetos de software que possibilite a extensão das metodologias de recomendação, técnicas de mineração e apresentação dos resultados. Isso facilitará a comunicação e a recomendação de indivíduos, bem como permitirá que os recursos de ranque, mineração e apresentação de resultados sejam modificados segundo as necessidades do projeto de software. O foco não é apresentar um novo método para qualificar esses especialistas, ou mesmo o uso das qualificações em si, mas sim a construção desta estrutura extensível que possibilite a agregação de várias metodologias para a identificação e recomendação de especialistas.

O restante do texto está estruturado como segue: no capítulo 2 são tratados os métodos utilizados para a elaboração deste trabalho; no capítulo 3 são apresentados detalhes da pesquisa de campo realizada; no capítulo 4 é discursado

sobre a fundamentação teórica e as tecnologias utilizadas; no capítulo 5 estão descritos os detalhes da execução e implementação do trabalho; no capítulo 6 são apresentados cenários de uso para este trabalho; e no capítulo 7 são descritas as conclusões e trabalhos futuros.

2 METODOLOGIA

O início deste trabalho deu-se pela leitura de diversos textos científicos da área de desenvolvimento distribuído de software. A partir dessa leitura foi identificada a existência de problemas relacionados à comunicação entre desenvolvedores em projetos distribuídos. Com o intuito de sanar algum desses problemas foram levantadas as possíveis dificuldades que poderiam ser atacadas. Assim, como objetivo para o presente trabalho foi escolhida a busca e recomendação de especialistas.

Para verificar o estado da arte do tema escolhido foi realizada uma revisão bibliográfica voltada à identificação de especialistas em projetos de desenvolvimento de software e sobre a comunicação entre membros de grupo de desenvolvimento distribuído. Detalhes desta revisão bibliográfica são apresentados na seção 4.1.

Após a revisão bibliográfica foi elaborado um questionário sobre as necessidades dos participantes de projetos de desenvolvimento distribuído de software. Ele foi aplicado entre os meses de março e abril de 2011 e envolveu apenas membros de projetos distribuídos. Maiores detalhes sobre o questionário e seus resultados são encontrados na seção 3.

Em seguida, foi definida uma arquitetura preliminar a ser utilizada. Para isso foram levantadas informações sobre a flexibilidade dos sistemas orientados a serviços e decidiu-se por utilizar esta abordagem na construção da arquitetura para este trabalho. Após a definição da abordagem a ser utilizada foram estudadas as tecnologias disponíveis para a comunicação e executados testes iniciais para confirmar sua utilidade e facilidade de uso. A seção 4.2 apresenta maiores informações sobre as decisões arquiteturais e as tecnologias utilizadas.

Após a escolha da tecnologia a ser utilizada, foi realizado um levantamento sobre os padrões de projeto a serem adotados durante o desenvolvimento. Os padrões foram escolhidos visando a extensibilidade e flexibilidade da aplicação. Os padrões escolhidos, bem como a justificativa sobre seu uso estão na seção 5.

Com os padrões de projeto e uma arquitetura orientada a serviços definidos, passou-se ao levantamento de informações do funcionamento e

construção de serviços web (*webservices*). Foram encontradas várias metodologias para construção destes serviços e realizado um estudo exploratório para determinar qual delas seria utilizada. Para execução deste estudo executou-se testes de implementação com os frameworks identificados para elaboração de serviços *web*. A metodologia adotada foi aquela que apresentou mais vantagens em relação à facilidade de uso. Maiores informações sobre a escolha da metodologia são encontradas na seção 4.2.

Após a escolha da tecnologia e metodologia a serem utilizadas foram definidas as interfaces e iniciada a construção do software para fornecer o serviço *web*. O passo seguinte foi a realização de um experimento para verificar o funcionamento e flexibilidade da arquitetura. Para isso foram criadas duas metodologias para obtenção de dados (mineradores) e três algoritmos para quantificar a *expertise* (qualificadores). Além disso, foram criados dois clientes para o serviço de qualificação: um portal *web* e um mensageiro instantâneo. Eles são capazes de utilizar as diferentes metodologias implementadas para recomendar especialistas.

3 PESQUISA DE CAMPO

Uma pesquisa foi conduzida com o intuito de identificar algumas necessidades dos desenvolvedores que atuam em projetos distribuídos. Ela foi realizada através de um questionário e tinha como um dos objetivos identificar as dificuldades dos desenvolvedores na coleta e interpretação de dados para identificação dos especialistas. Esta pesquisa foi realizada nos meses de março e abril de 2011 e envolveu apenas membros de projetos de software distribuídos. Seus participantes possuem experiência entre 3 e 30 anos na área de desenvolvimento de software e 1 a 10 anos com desenvolvimento distribuído.

O questionário (disponível no apêndice A) foi dividido em duas seções: a primeira tem por objetivo obter informações sobre o perfil do respondente; a segunda possui questões relativas ao desenvolvimento distribuído. Nesta segunda seção questionou-se sobre quais os meios utilizados para identificar especialistas e quais informações os desenvolvedores consideram importantes durante seu trabalho. Para responder estas questões foram apresentadas dez informações aos respondentes e eles podiam atribuir pesos de 0 a 5 a cada uma delas (sendo 0: não tem importância; e 5: essencial). Mediante os resultados obtidos, verificou-se que as três informações consideradas mais importantes, foram, na ordem:

1. Qual desenvolvedor é o especialista (*expert*) de um dado artefato? (média 4,065);
2. De quais outros artefatos um dado artefato depende? (média 4,032)
3. Quais outros artefatos dependem de um dado artefato? (média 3,968)

Este resultado demonstra a grande necessidade dos desenvolvedores com relação à obtenção de informações sobre quem pode oferecer ajuda em um dado trecho de código (item 1). Foi considerada importante também a possibilidade de verificação de dependência lógica e estrutural entre artefatos (itens 2 e 3). Entretanto, não trataremos destes dois últimos itens no escopo deste trabalho.

A partir dos dados coletados pôde-se ainda verificar que, quando existe a necessidade de identificar especialistas, 62% dos entrevistados buscam por

informações no sistema de gerenciamento de versões, fóruns ou *mail threads*. Isto demonstra a riqueza das informações que podem ser mineradas e utilizadas para localização de especialistas.

Os participantes também foram questionados sobre quais meios eles mais utilizam para identificação dos especialistas. As respostas recebidas variam entre: procura manual em fóruns; procura manual na documentação do projeto; contato com pessoas já conhecidas; contato com o responsável pelo projeto e busca no histórico do repositório de códigos fontes. Cabe salientar que, alguns dos participantes afirmaram ainda utilizar mais de um destes métodos.

Eles ainda foram questionados sobre como se sentem iniciando uma conversa com pessoas que não conhecem. Alguns deles afirmaram não se sentir confortáveis nessas situações e um agravante para esta dificuldade foi identificado quando os participantes da pesquisa informaram que trabalham com pessoas de países diferentes países (Brasil, Índia, Itália, Estados Unidos e Canadá). Por esse motivo, as diferenças culturais e de fuso-horários devem ser consideradas na comunicação. Dadas estas dificuldades, uma aplicação que promova apenas um cliente de mensagens síncronas apresenta-se insuficiente porque alguns membros do time podem não estar conectados quando outro membro precisar de ajuda.

Em resumo, a partir dos resultados obtidos foi possível identificar quatro requisitos desejáveis para uma ferramenta que auxilie na identificação dos especialistas no contexto do desenvolvimento distribuído de software:

- **R1.** Não existe senso comum sobre a melhor fonte de pesquisa para identificar um especialista. Isso depende das necessidades do desenvolvedor, dos recursos disponíveis e do contexto do projeto;
- **R2.** Existem diversos métodos (algoritmos) para recomendar especialistas e quantificar a *expertise* de um indivíduo. E isso inclui as diferentes fontes de dados passíveis de serem utilizadas (**R1**);
- **R3.** Alguns desenvolvedores se sentem mais confortáveis do que outros quando precisam trocar informações com indivíduos que não conhecem;
- **R4.** Membros de times globais estão localizados em diferentes países. Isso resulta em diferentes culturas e fuso-horários. Então o cliente que exibe os resultados da busca precisa se adaptar a essas características.

A partir destas informações foi realizada uma revisão bibliográfica para

identificar as ferramentas existentes que teve por objetivo analisar o modo como as ferramentas tratam os requisitos expostos pela pesquisa e quais desses requisitos são contemplados. Detalhes da revisão bibliográfica são apresentados na seção 4.1.

4 REFERENCIAL TEÓRICO

Neste capítulo será apresentada a revisão da literatura realizada sobre a localização de especialistas em projetos de software (seção 4.1). Será apresentada também uma breve discussão relacionada às tecnologias que foram utilizadas no desenvolvimento deste trabalho (seção 4.2).

4.1 REVISÃO BIBLIOGRÁFICA

Várias ferramentas e pesquisas têm sido desenvolvidas para facilitar a comunicação entre os membros de equipes distribuídas (DULLEMOND; VAN GAMEREN; VAN SOLINGEN, 2010), (MOCKUS; HERBSLEB, 2002), (KWAN; DAMIAN; STOREY, 2006), (SHAMI; EHRLICH; MILLEN, 2006). Estas ferramentas possibilitam não apenas a comunicação entre os indivíduos do grupo, mas também permitem identificar os especialistas da equipe e facilitar o acesso às habilidades deles entre os membros de equipes dispersas (MOCKUS; HERBSLEB, 2002), (SHAMI; EHRLICH; MILLEN, 2006), (BEGEL; DELINE; ZIMMERMANN, 2010), (TRINDADE; MORAES; BARBOSA, 2009), (MINTO; MURPHY, 2010).

Expertise Browser (ExB) (MOCKUS; HERBSLEB, 2002) é uma ferramenta que utiliza a idéia de EAs (*Experience Atoms* – Átomos de experiência), que são unidades de experiência obtidas a partir do repositório de códigos fontes e de gerenciadores de conteúdo (*Content Management Systems* - CMS). As EAs são utilizadas para determinar os especialistas nas várias partes do projeto. A ExB traça a rede de relacionamentos entre artefatos, pessoas, tarefas e funcionalidades. Sua finalidade é promover a localização dos especialistas e traçar seus relacionamentos em um ambiente web.

SmallBlue (SHAMI; EHRLICH; MILLEN, 2006) é uma ferramenta geradora de redes sociais sobre projetos de software. Ela utiliza análises de e-mails e mensagens instantâneas entre os desenvolvedores para determinar os *rankings* de *expertise*. Essas análises são realizadas associando nomes e tópicos extraídos dos textos das mensagens. A busca por especialistas é feita por meio de uma página

web, dado um determinado assunto, habilidade, biblioteca ou módulo fornecido pelo usuário e os resultados são apresentados numa lista ordenada que contenha os especialistas naquele assunto.

Codebook (BEGEL; DELINE; ZIMMERMANN, 2010) é uma ferramenta que gera uma rede social a partir dos repositórios de códigos fontes, mensagens e documentação. São estabelecidos vínculos entre as atividades (*item work*), seus artefatos e desenvolvedores. A rede resultante pode ser consumida por ferramentas auxiliares como o Hoozizat, um portal web específico para busca de especialistas em funcionalidades, APIs, produtos ou sistemas.

Presley (TRINDADE; MORAES; BARBOSA, 2009) utiliza o histórico dos códigos fontes e no registro da comunicação entre membros dos times. Seu objetivo é facilitar a comunicação de um dado projeto de software, recomendando especialistas por áreas de domínio. A qualificação destes especialistas se dá pelo histórico de alterações e registros de comunicação entre os desenvolvedores. Presley utiliza-se de uma metodologia de recomendação de especialistas criada especificamente para a ferramenta, que não permite modificações.

Emergent Expertise Locator (EEL) (MINTO; MURPHY, 2010) utiliza o histórico de alterações dos códigos fontes para qualificar os especialistas dos artefatos. A ferramenta faz uso da matriz de *coordination requirements* proposta por Cataldo (CATALDO; HERBSLEB; CARLEY, 2008) para ranquear os especialistas. A EEL possibilita também a troca de mensagens síncronas e assíncronas entre os membros da equipe. Porém, assim como o trabalho anterior, o EEL possui uma estrutura rígida de recomendação de especialistas.

Todos os trabalhos acima citados proveem a identificação de especialistas entre os membros da equipe distribuída. Cada uma dessas ferramentas utiliza sua própria metodologia de identificação de especialistas e mineração de dados. Elas não fornecem meios para extensão ou adição de novos recursos porque foram criadas para contextos e necessidades específicas. Além disso, os clientes utilizados pelas ferramentas são estáticos. Estes clientes não objetivam a facilidade de extensões e alterações. Não sendo possível sua alteração e reconstrução para atender necessidades específicas dos projetos de software.

É possível observar ainda que, não há consenso sobre os modos de identificação de especialistas nem sobre a metodologia empregada nas diferentes ferramentas existentes. O foco do presente trabalho é justamente projetar uma

arquitetura extensível que possibilite o uso de uma ou várias metodologias que melhor convierem ao projeto de software e seus usuários. E que possibilite o desenvolvimento e extensão de outros aplicativos consumidores (clientes) para os serviços de qualificação de especialistas.

4.2 TECNOLOGIAS UTILIZADAS

Nesta seção, serão apresentadas as tecnologias, metodologias e padrões utilizados no desenvolvimento da solução proposta, visando promover flexibilidade das metodologias de recomendação, técnicas de mineração e da apresentação dos resultados de buscas.

Para possibilitar esta flexibilidade, adotou-se o desenvolvimento com amplo uso de interfaces, bem como alguns padrões de projeto. E, para alcançar a independência de plataforma, optou-se por sua construção utilizando a linguagem Java. Com essa linguagem é possível desenvolver a conexão entre computadores em três abordagens distintas: RMI (*Remote Method Invocation*), serviços web e conexão via *socket*.

A abordagem RMI determina que tanto o lado cliente quanto o lado servidor da aplicação sejam, obrigatoriamente, desenvolvidos na linguagem Java. Esta abordagem é utilizada na construção da ferramenta Presley (TRINDADE; MORAES; BARBOSA, 2009). Isto torna a aplicação cliente totalmente dependente da linguagem Java, impedindo que clientes em ambientes que não suportem esta linguagem sejam implementados.

Ao contrário da RMI, a conexão via *socket* é independente da linguagem de programação utilizada. Assim, apesar da aplicação servidora ser implementada em Java, a aplicação cliente pode ser implementada em qualquer linguagem que dê suporte a *sockets*. Entretanto, ambas as aplicações (cliente e servidor) precisam estar de comum acordo quanto às regras para o tráfego de requisições e respostas. A utilização de sockets obriga a definição de um protocolo de comunicação particular para que tanto cliente quanto servidor possam interpretar os dados trafegados pela rede.

Outra possível abordagem é o uso de serviços *web* (*webservices*). Essa

tecnologia surgiu da necessidade de integrar sistemas heterogêneos, possibilitando a independência tanto de linguagem quanto de plataforma (GOMES, 2009). Esta é a abordagem escolhida para a execução deste trabalho. Escolhida não apenas por sua independência de plataforma e de linguagem, mas também por possuir um padrão de comunicação definido internacionalmente. O uso do protocolo SOAP (*Simple Object Access Protocol*) dispensa a definição de um protocolo particular ao contrário da abordagem utilizando *sockets*.

O desenvolvimento de serviços *web* com a linguagem Java pode ser realizado através de *frameworks*, dos quais se destacam: Apache Axis 2, JAX-WS 2.1 e o conjunto Apache CXF 2.3.7 com Spring-WS 1.5.9. Estes *frameworks* foram experimentados para verificar qual deles apresentaria menos dificuldades durante a implementação.

O Apache AXIS 2 foi rejeitado inicialmente devido à obrigatoriedade de a aplicação *web* AXIS 2 ser implantada no servidor de aplicação para prover a publicação e execução dos serviços. Esta característica é dispensada pelos demais.

Nos *frameworks* remanescentes foram identificados dois modos de operação:

- serviço por geração dinâmica - Para geração dinâmica de serviços web são utilizadas anotações (*Annotations*) e a construção é determinada a partir da implementação corrente.
- serviço baseado em contratos - Um serviço *web* baseado em contrato não terá suas assinaturas de mensagens afetadas por alterações no software uma vez que estas assinaturas são definidas por arquivos WSDL (*Web Service Description Language*) gerados antes da implementação do serviço.

A característica da construção dinâmica pode ser uma desvantagem em aplicações em que a implementação do serviço web mude com frequência. Uma vez que todos os clientes do serviço *web* precisam estar cientes da estrutura da mensagem corrente. A mudança dessa estrutura pode acarretar modificações em todos os clientes para que a troca de mensagens seja possível.

O uso de contratos obriga a criação dos arquivos WSDL antes da construção do serviço. Estes arquivos definem a assinatura das mensagens de requisição e resposta dos serviços *web*. Isso significa que mudanças na

implementação não podem afetar a estrutura das mensagens. Mas nos casos onde sejam necessárias mudanças na assinatura para inclusão ou remoção de valores tanto o WSDL como a implementação do serviço deverá ser modificada.

Foi verificado que o Spring-WS com Apache CXF e o JAX-WS permitem o uso de ambas as abordagens. Mas o uso do Spring-WS e do Apache CXF estão condicionados à inserção de bibliotecas de terceiros no servidor de aplicações. Enquanto que o JAX-WS conta com a especificação e implementações no próprio *kit* de desenvolvimento da linguagem Java. Apesar disso, não foram identificadas diferenças consideráveis na facilidade de uso entre as bibliotecas.

Para reduzir o uso de bibliotecas de terceiros foi adotado o JAX-WS com a construção dinâmica como modo de operação. Foi adotado esse tipo de construção, pois a arquitetura aqui proposta não prevê mudanças na assinatura das mensagens e, além disso, a construção dinâmica dispensa o tempo demandado com a definição do WSDL que define o serviço.

5 FERRAMENTA PROPOSTA

Conforme apontado na seção 4.1, existem vários modos para localizar e ranquear os especialistas em projetos de software. Esta recomendação pode ser realizada utilizando informações provenientes de sistemas de gerenciamento de versão, listas de discussão, repositórios de códigos fontes, documentos, entre outros (MOCKUS; HERBSLEB, 2002), (SHAMI; EHRLICH; MILLEN, 2006), (MINTO; MURPHY, 2010). Essa procura por especialistas pode consumir uma quantidade significativa do tempo dos desenvolvedores. A fim de reduzir este tempo de busca foram propostas várias ferramentas, cada uma para fins e objetivos específicos.

Este trabalho diferencia-se dos demais, pois apresenta uma arquitetura extensível, que possibilita o uso da metodologia de recomendação que melhor se adapte às necessidades dos projetos. Para promover a flexibilidade e escalabilidade das metodologias de busca foi adotada uma arquitetura em três camadas utilizando tecnologias de serviços *web* e baseada em interfaces e padrões de projeto. Esta decisão foi tomada porque, de acordo com (JOHNSON *et al.*, 2005), a simples utilização do paradigma orientado a objetos não é suficiente para garantir a flexibilidade de um software.

Os detalhes da arquitetura e suas características de extensibilidade serão apresentados a seguir.

5.1 ARQUITETURA DA APLICAÇÃO SERVIDORA

Como dito anteriormente, a arquitetura aqui proposta está divida em três camadas, a saber: camada de comunicação, camada de qualificação e camada de mineração. Apesar da divisão, as três camadas operam em conjunto para atender às requisições dos clientes. A figura 1 ilustra esta divisão em camadas.

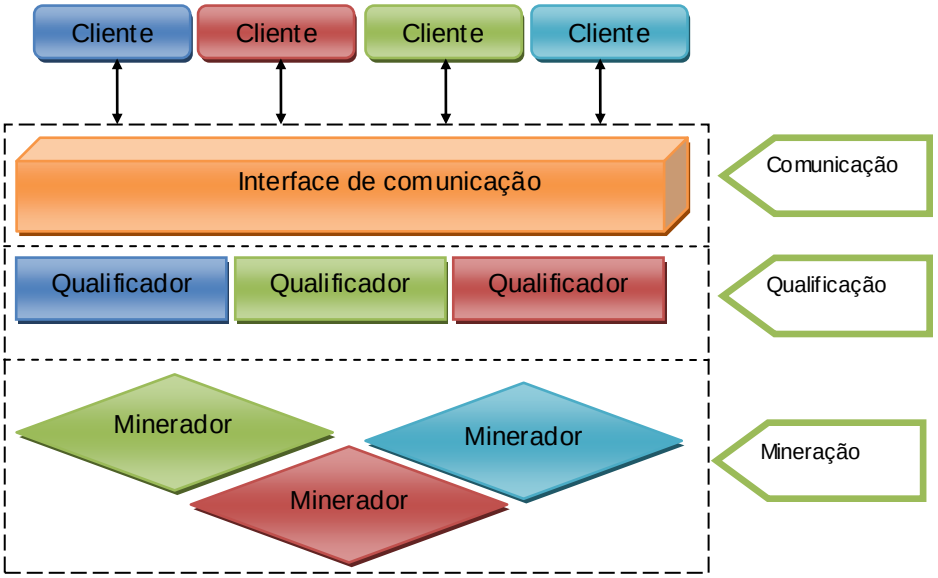

Figura 1 - Camadas da arquitetura desenvolvida neste trabalho

Fonte: Autoria própria

A camada de comunicação objetiva prover baixo acoplamento entre clientes e aplicação servidora. Essa camada é a via pela qual os clientes e as metodologias para qualificação de especialistas interagem.

A camada de mineração é formada por objetos do tipo `Miner` (Mineradores). Eles possibilitam recuperar dados a partir de diferentes fontes de dados (**R1**). Essa obtenção pode ser realizada a partir de arquivos de texto, repositórios de códigos fontes, fóruns de discussão, entre outros. Os dados obtidos são utilizados pelas demais entidades da aplicação para executar a identificação dos especialistas.

A camada de qualificação é formada por objetos do tipo `Qualifier` (Qualificador). Eles são objetos que utilizam os dados dos mineradores para determinar a *expertise* de cada desenvolvedor. Eles possuem diferentes algoritmos de quantificação da *expertise* e os aplicam sobre os dados recuperados pelos mineradores (**R2**).

5.1.1 A CAMADA DE COMUNICAÇÃO

Para desacoplar os clientes da aplicação servidora decidiu-se pela sua implementação na forma de um serviço *web*. A razão para esta escolha é que uma arquitetura orientada a serviços auxilia a manutenção da flexibilidade e escalabilidade de sistemas (JOSUTTIS, 2007). O uso de serviços possibilita a criação de clientes por terceiros que possam acessar as metodologias disponíveis, adaptando a apresentação dos resultados às suas necessidades e/ou contexto de seus próprios projetos.

A camada de comunicação disponibiliza dois serviços: um para listagem das metodologias de recomendação disponíveis e outro para executar a identificação (listagem) dos especialistas em um assunto. A listagem das metodologias como um serviço possibilita que clientes já existentes façam uso de novos métodos de recomendação sem a necessidade de alteração de seu código. A figura 2 apresenta a camada de comunicação e seus serviços.

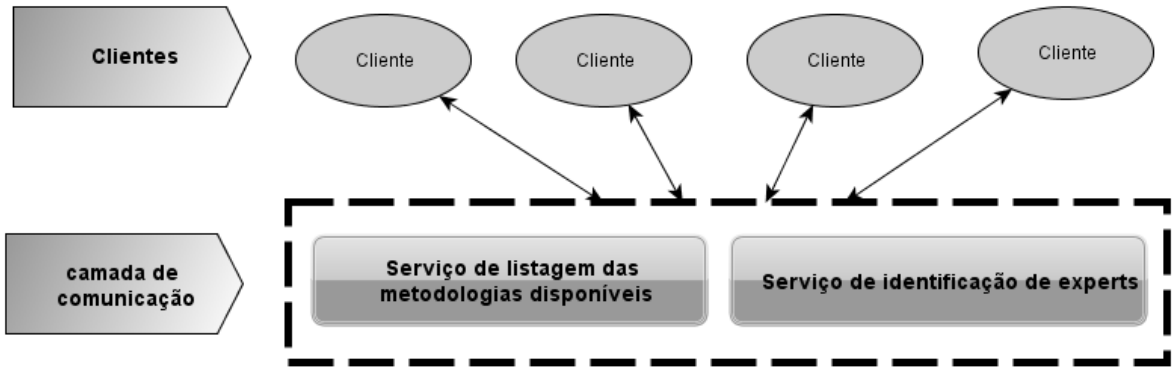

Figura 2- Ilustração da camada de comunicação

Fonte: Autoria própria

5.1.2 AS CAMADAS DE QUALIFICAÇÃO E MINERAÇÃO

A camada de mineração contém as metodologias para acesso às fontes

de dados e a camada de qualificação contém os algoritmos que quantificam a *expertise* dos desenvolvedores. Essas camadas contêm os objetos mineradores e qualificadores, respectivamente:

- objetos mineradores: são objetos capazes de obter dados a partir uma dada fonte. A obtenção destes dados pode ser realizada a partir de arquivos de texto, repositórios de códigos fontes ou fóruns de discussão. Os dados obtidos por estes objetos são utilizados pelas demais entidades da aplicação para executar a identificação dos especialistas;
- objetos qualificadores: são objetos que utilizam os dados dos mineradores para determinar a *expertise* de cada desenvolvedor. Eles possuem algoritmos para processamento dos dados recuperados pelos mineradores. Estes algoritmos extraem o coeficiente de *expertise* e criam tuplas (desenvolvedor, coeficiente de *expertise*) enviando esses valores como respostas para a solicitação do cliente.

Tanto os qualificadores quanto os mineradores possibilitam a extensibilidade das metodologias de recomendação da aplicação, visto que eles não precisam ser parte do núcleo da aplicação servidora e podem ser adicionados como *plugins* da aplicação. Em outras palavras, a arquitetura do serviço foi criada como uma infraestrutura de componentes, permitindo que novos mineradores e qualificadores sejam criados e adicionados à aplicação sem a necessidade de alteração ou nova compilação do núcleo da aplicação.

Cada qualificador existente deve estar vinculado a um minerador existente. Internamente, o qualificador repassa os parâmetros de busca para o minerador. O qualificador recebe, então, os dados adquiridos pelo minerador e aplica seu algoritmo de qualificação sobre estes dados. Esses algoritmos extraem o coeficiente de *expertise* e criam tuplas (desenvolvedor, coeficiente de *expertise*) enviando esses valores para a camada de comunicação que os envia como resposta para a requisição do cliente. Os detalhes da implementação e funcionamento da arquitetura serão apresentados nas próximas seções.

5.2 FUNCIONAMENTO DA ARQUITETURA

Nesta seção será apresentado o funcionamento da arquitetura proposta para busca e recomendação de especialistas, mostrando o fluxo de execução, os relacionamentos entre as camadas e seus detalhes internos.

Todo o processo inicia-se pela requisição de identificação de especialistas proveniente de um cliente. Esta requisição deve escolher a metodologia de busca a ser utilizada. Para que os clientes saibam quais mecanismos são providos pela arquitetura no momento, foi implementado um serviço que fornece uma lista das metodologias de identificação de especialistas.

Tendo conhecimento das metodologias de busca disponíveis é possível enviar uma requisição para um destes métodos através do serviço de identificação dos especialistas. Mesmo que existam várias implementações específicas de metodologias de identificação, a aplicação no formato de serviço web permite que seja utilizado um único canal de entrada e saída para as requisições dos clientes. Essas requisições são enviadas para o serviço por intermédio de uma mensagem XML. A aplicação web utiliza as funcionalidades do *framework* JAX-WS para converter essas mensagens em objetos específicos da implementação. Assim quando uma requisição é enviada ao serviço web a mensagem é transformada em um objeto da classe `Bean`. A estrutura da classe `Bean` pode ser vista na figura 3.

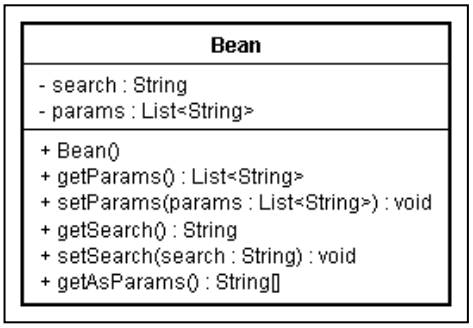

Figura 3 – Estrutura da classe do objeto Bean.
Fonte: Autoria própria

O objeto `Bean` possui dois atributos: uma lista de parâmetros; e um nome de busca (`search`). A lista de `String` contém termos ou palavras chaves que serão utilizados pelos mineradores para restringir o conjunto de dados recuperados. E o nome da busca identifica o par de objetos (minerador, qualificador) que atenderá à solicitação do cliente. Os valores destes atributos são definidos diretamente pela JAX-WS. E estes são os dados fundamentais utilizados pelo serviço para atendimento de uma requisição para identificação de especialistas.

Estes valores são enviados para o objeto `SearchEngine` que realiza as operações necessárias para criação dos mineradores e qualificadores que correspondem à busca solicitada. Após realizar a instanciação destes objetos o `SearchEngine` envia a lista de `String` como parâmetros para os mineradores e qualificadores. Estes por sua vez contém as implementações específicas da metodologia para identificação dos especialistas. A figura 4 ilustra esta execução na forma de diagrama de sequência de execução de uma busca por especialistas.

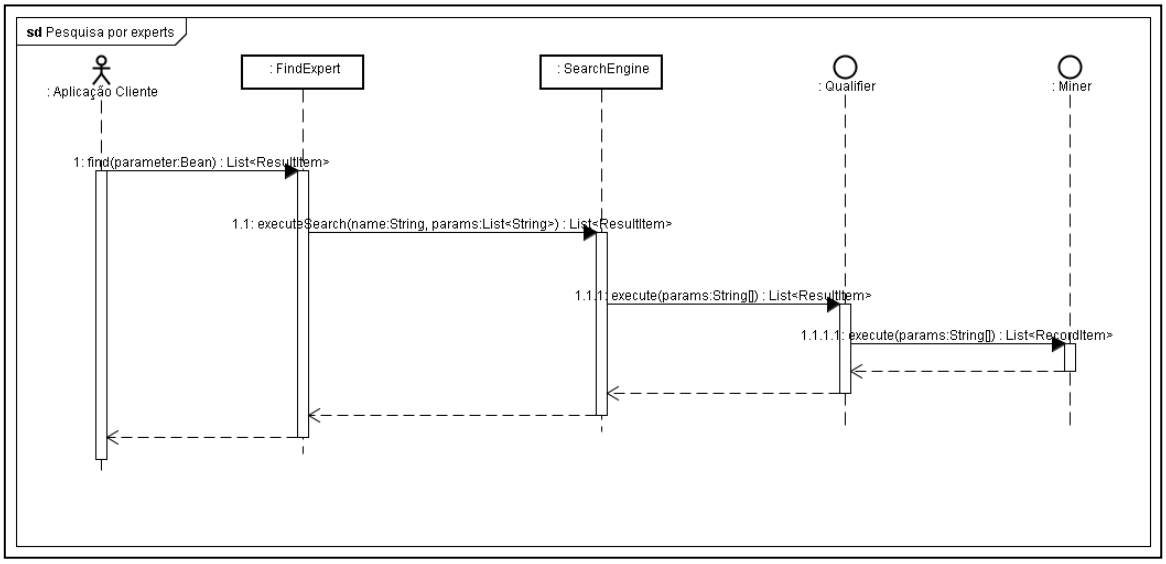

Figura 4 – Diagrama de sequência da execução de uma busca no serviço.
Fonte: Autoria própria

Uma vez que a `SearchEngine` envia o comando de execução para um qualificador este solicita ao minerador que recupere os dados utilizando a lista de

`String` para reduzir o escopo da busca. O resultado deste levantamento é transformado numa lista de objetos do tipo `RecordItem` onde cada linha de dado recuperada pelo minerador é transformada em um item da lista. A estrutura da classe desse objeto pode ser visualizada na figura 5.

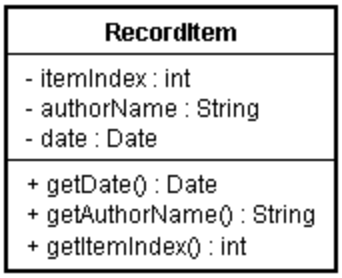

Figura 5 – Estrutura da classe que contém os dados de um registro encontrado pelo objeto minerador.
Fonte: Autoria própria

O objeto do tipo `RecordItem` contém um índice, um autor e uma data. O índice (`itemIndex`) pode ser utilizado para ordenação porque apresenta unicamente a posição que o registro ocupa nos registros levantados pelo minerador. O nome de autor (`authorName`) indica o membro do grupo de desenvolvimento que foi responsável pelo evento identificado pelo minerador. E a data (`date`) representa a data do evento disparado pelo autor correspondente.

A partir dessa lista de `RecordItems` o objeto qualificador executa seus algoritmos específicos para quantificar o coeficiente de *expertise* de cada um dos autores existentes nos objetos desta. Essa execução gera uma lista de objetos `ResultItem` que será devolvida à `SearchEngine` como sendo o *ranking* dos especialistas referente à requisição do cliente. Cada objeto `ResultItem` contém um nome de autor (`name`) e um coeficiente de *expertise* (`coefficient`). A estrutura desse objeto é apresentada na figura 6.

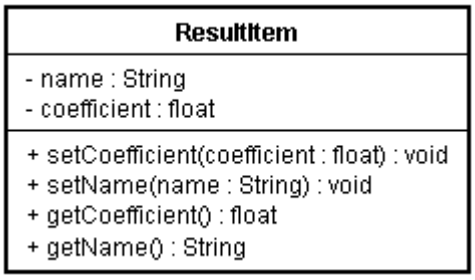

Figura 6 – Estrutura da classe que contém os dados processados pelo qualificador.
Fonte: Autoria própria

A `SearchEngine` que envia a lista de resultados como resposta ao cliente na forma de uma mensagem XML. A conversão da lista de objetos para mensagem XML também é realizada pela JAX-WS, assim a `SearchEngine` não precisa operar diretamente na conversão destes dados.

Durante a execução da aplicação web vários clientes podem requisitar a recomendação de especialistas simultaneamente. Para evitar a reconstrução de todos os objetos à cada requisição decidiu-se pela implementação do objeto `SearchEngine` como um `Singleton` (KUCHANA, 2004).

Para atendimento de uma requisição, o objeto `SearchEngine` recebe como parâmetros de execução um nome de busca e a lista dos parâmetros. Ele utiliza esse nome para localizar uma configuração entre as existentes em sua lista de buscas (`Search`) pré-carregadas. A figura 7 apresenta a estrutura da classe `SearchEngine` utilizada para a execução das buscas por especialistas.

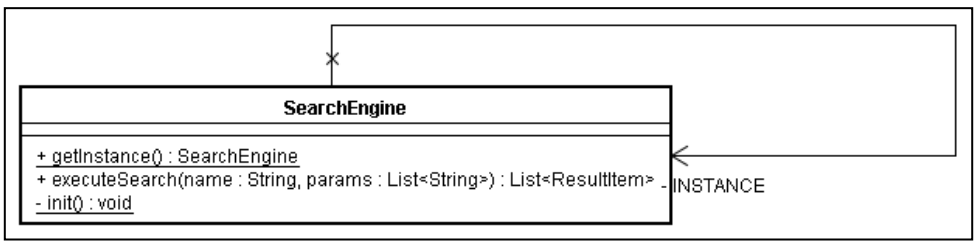

Figura 7 – Estrutura da classe SearchEngine que atende às requisições recebidas pelo serviço web.
Fonte: Autoria própria

O objeto `SearchEngine` procura pelo nome solicitado e recupera as informações necessárias à execução do restante do processo para atender à requisição do cliente. Cada um destes objetos do tipo Search possui como atributos: um apelido (`alias`) que é utilizado para identificação da busca; um apelido de minerador (`minerAlias`) que identifica qual implementação do minerador é utilizado para esta busca; e um apelido de qualificador (`qualifierAlias`) que identifica qual implementação específica dos qualificadores atenderá a requisição. A figura 8 ilustra a estrutura da classe `Search`.

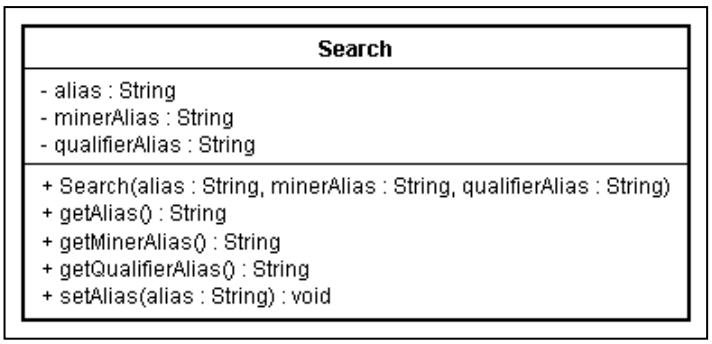

Figura 8 – Estrutura da classe do objeto Search.
Fonte: Autoria própria

O objeto `SearchEngine` contém ainda dois localizadores de entidades. Um localizador de mineradores (`Locator<Miner>`) e um localizador de qualificadores (`Locator<Qualifier>`). A partir destes localizadores é possível solicitar a instanciação dinâmica de um objeto por seu apelido. Deste modo o `Singleton` utiliza os apelidos existentes no objeto `Search` para instanciar os devidos minerador e qualificador da respectiva busca por intermédio do localizador. A figura 9 apresenta a estrutura da classe dos localizadores.

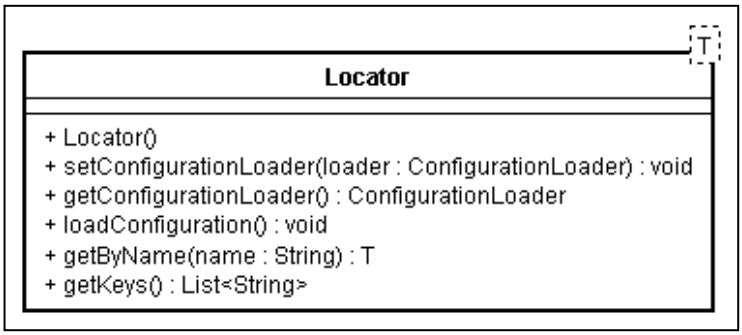

Figura 9 – Estrutura da classe do objeto Locator.
Fonte: Autoria própria

Os localizadores são objetos que utilizam referências genéricas (`Generics`) para valer-se de uma mesma implementação para diversos objetos evitando a duplicação de código. O uso de referências genéricas é um dos recursos apresentados pela linguagem Java para ampliar o reúso de código, uma vez que a referência `<T>` pode ser qualquer objeto existente no contexto do software. Os localizadores constroem sua lista de objetos baseados nas configurações lidas pelos objetos do tipo carregador de configuração (`ConfigurationLoader`). Os carregadores permitem que as configurações externas sejam entendidas pela aplicação web. Eles também foram construídos para operar com referências genéricas. A figura 10 apresenta a assinatura da interface que define os objetos do tipo `ConfigurationLoader`.

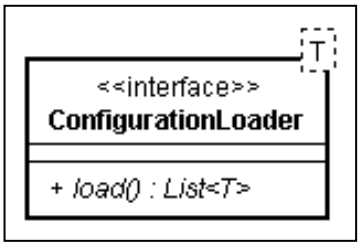

Figura 10 – Estrutura da interface ConfigurationLoader.
Fonte: Autoria própria

Uma vez identificados e carregados os objetos mineradores e qualificadores são enviados de volta para a `SearchEngine` para que a execução da

solicitação seja realizada. Tanto os mineradores quanto qualificadores são construções do padrão de projeto `Command`. Esse padrão possibilita que diferentes implementações sejam utilizadas com baixo acoplamento estático e impede que, durante o processamento de uma requisição, o cliente ou mesmo o `Singleton` tome consciência do objeto concreto que realiza a busca pelos especialistas porque a execução é delegada a uma implementação particular (JOHNSON *et al.*, 2005). Esses objetos possuem interfaces com as assinaturas fundamentais para que um artefato seja considerado um minerador ou um qualificador. A figura 11 apresenta as interfaces destes dois elementos.

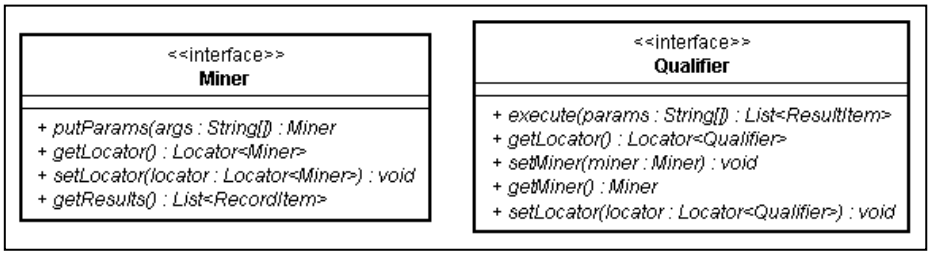

Figura 11 - Interfaces dos mineradores à esquerda, e dos qualificadores à direita.
Fonte: Autoria própria

Os objetos `Miner` e `Qualifier` recebem vetores de `String` como parâmetros para a execução das buscas e são as principais áreas de extensão desta aplicação *web* porque é a partir destas interfaces que são definidas as metodologias de busca por especialistas. Cada busca (`Search`) contém os apelidos de um minerador e de um qualificador. Mas este uso não é exaustivo, um mesmo minerador pode ser utilizado em conjunto com outros qualificadores desde que exista uma instância de `Search` que realize esta ligação. Assim além de alcançar a extensibilidade com as implementações específicas ainda é possível o reúso de mineradores ou qualificadores já existentes para modelar novas metodologias de busca.

O objeto `SearchEngine` (Figura 7) utiliza a instância da `Search` para obter os mineradores e qualificadores configurados para uma dada metodologia de recomendação de especialistas e repassa à eles os parâmetros enviados pelo cliente na requisição. Após sua execução, os qualificadores retornam os resultados

para a `SearchEngine` que envia este resultado como saída do seu método `executeSearch`. O retorno deste método contém uma lista de resultados (Figura 6).

Como já foi dito, a lista dos resultados será enviada como resposta ao cliente que fez a requisição que pode, então, apresentá-la da maneira que melhor lhe convier.

5.3 A EXTENSIBILIDADE E FLEXIBILIDADE POR INTERMÉDIO DE CONFIGURAÇÕES E INTERFACES

A tipagem dos mineradores e qualificadores foi definida por um conjunto de interfaces em separado. Este conjunto de interfaces foi movido do projeto principal para uma biblioteca. Isso permite que terceiros utilizem esse conjunto de interfaces em tempo de compilação para a criação das bibliotecas que serão adicionadas na aplicação servidora.

A arquitetura carrega todos os mineradores e qualificadores existentes em seu `CLASSPATH` e que estejam habilitados nos arquivos de configuração. São três os arquivos que configuram as metodologias de identificação de especialistas: `webminers.properties`, `webqualifiers.properties` e `websearches.xml`. Eles estão localizados no diretório de nome `config` na raiz da aplicação web.

Os arquivos `webminers.properties` e `webqualifiers.properties` são arquivos texto plano contendo entradas contendo tuplas no formato `CHAVE=VALOR`. Nestes arquivos cada linha corresponde a uma tupla. A chave é o apelido que será utilizado para identificação do minerador/qualificador no interior da aplicação *web* e o valor é o nome totalmente qualificado (nome da classe e caminho do pacote onde está localizada) da classe que é identificada por aquele apelido. As figuras 12 e 13 apresentam exemplos do conteúdo desses arquivos.

```
aliasQualifier1=nomeDaClasseDeQualificação1TotalmenteQualificado
```

```
aliasQualifier2=nomeDaClasseDeQualificação2TotalmenteQualificado
```

Figura 12 - Exemplo de conteúdo do arquivo webqualifiers.properties.

Fonte: Autoria própria

```
aliasMiner1=nomeDaClasseDeMineração1TotalmenteQualificado
aliasMiner2=nomeDaClasseDeMineração2TotalmenteQualificado
```

Figura 13 - Exemplo de conteúdo do arquivo webminers.properties.

Fonte: Autoria própria

Já o arquivo `websearches.xml` é um arquivo XML que contém a configuração das buscas e seus apelidos. A estrutura deste é apresentada na figura 14. Nesse arquivo, cada *tag* `<search>` define uma configuração na aplicação *web*. A configuração determina o apelido da busca (`<alias>`), o apelido do minerador (`<miner>`) e o apelido do qualificador (`<qualifier>`) que será utilizado. Os apelidos existentes nestes três arquivos devem estar consistentes entre si para evitar falhas de referência durante o atendimento da requisição pelo `Singleton` (Figura 8).

```
<?xml version="1.0" encoding="UTF-8"?>
<searches>
      <search>
        <alias>Alias da busca 1</alias>
        <qualifier>aliasQualifier1</qualifier>
        <miner>aliasMiner1</miner>
    </search>
      <search>
        <alias>Alias da busca 2</alias>
        <qualifier>aliasQualifier2</qualifier>
        <miner>aliasMiner2</miner>
     </search>
      <search>
        <alias>Alias da busca 3</alias>
        <qualifier>aliasQualifier1</qualifier>
        <miner>aliasMiner2</miner>
     </search>
```

```
    <search>
        <alias>...</alias>
        <qualifier>...</qualifier>
        <miner>...</miner>
    </search>
      .
      .
      .
</searches>
```

Figura 14 - Estrutura do arquivo XML que configura as metodologias de busca disponíveis.

Fonte: Autoria própria

Os `Miners` e `Qualifiers` são gerenciados por localizadores (`Locators`) que possibilitam a identificação e carga dinâmica das implementações específicas solicitadas durante a execução de uma requisição do cliente. Para carregar os arquivos de configuração os localizadores utilizam implementações da interface `ConfigurationLoader`. Existem duas implementações para leitura de configurações: `ConfigurationLoaderProperties` e `ConfigurationLoaderXML`. A figura 15 apresenta a estrutura da classe que lê os arquivos de `properties` e da classe que lê o arquivo XML e que repassam essas configurações aos localizadores.

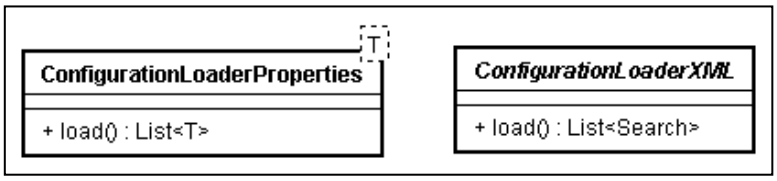

Figura 15 - Estrutura da classe de configuração de arquivos properties à esquerda e da classe de configuração do arquivo XML à direita.

Fonte: Autoria própria

5.4 ATENDENDO AOS REQUISITOS

O objetivo da aplicação construída neste trabalho é alcançar os requisitos identificados na seção 3. Os requisitos **R1** e **R2** são resolvidos com o uso dos elementos apresentados na seção 5.1.2, já que estes permitem que diferentes metodologias de buscas sejam utilizadas pela aplicação. Sendo assim cada interessado pode alterar a metodologia de busca de especialistas da maneira que melhor se adequar às suas necessidades. Eles podem utilizar diferentes fontes de dados, mineradores e qualificadores já existentes ou criar suas próprias implementações para atender às requisições de identificação de especialistas.

Já os requisitos **R3** e **R4** são obtidos uma vez que a aplicação foi construída na forma de serviço web, desacoplando os clientes da aplicação e permitindo diferentes maneiras de apresentar os resultados. Com isso é possível criar de clientes com características que contornem as dificuldades geradas pelas diferenças pessoais, culturais e temporais.

6 CENÁRIOS DE USO

Para verificar a flexibilidade e extensibilidade da arquitetura proposta foram criadas implementações de *plugins*: dois mineradores e três qualificadores. E, para verificar o funcionamento do serviço web, foram construídas duas aplicações clientes que podem também alternar entre qual das metodologias de identificação de especialistas entre as disponíveis será utilizada. As implementações dos *plugins* são apresentadas na figura 16, essas implementações serão detalhas a seguir.

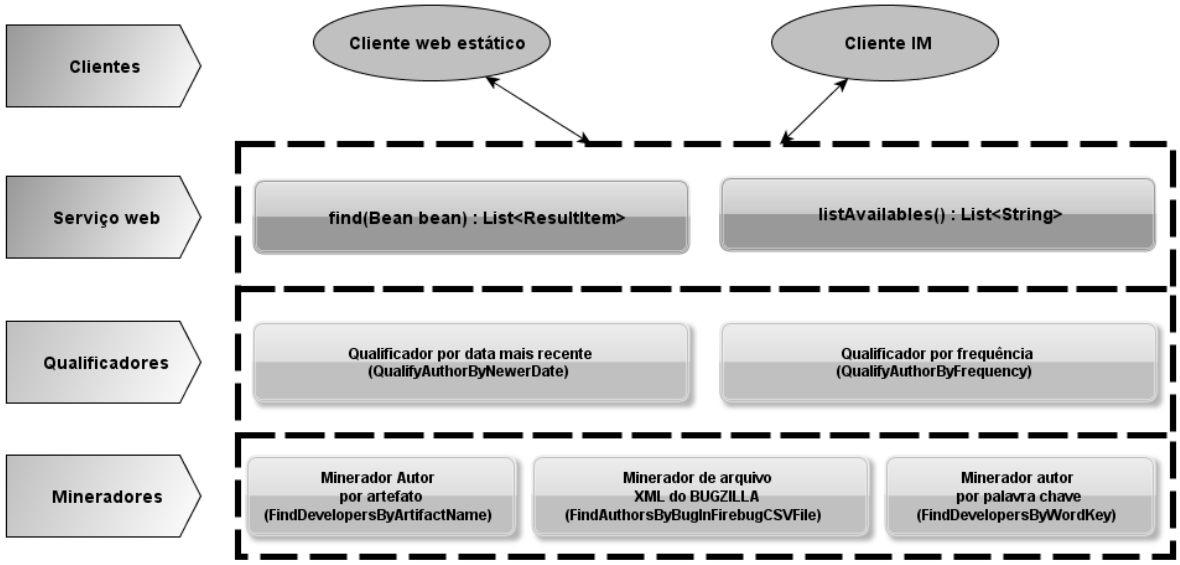

Figura 16 - Representação da arquitetura com clientes, serviços, mineradores e qualificadores instanciados.

Fonte: Autoria própria

6.1 METODOLOGIAS DE BUSCAS IMPLEMENTADAS

Foram implementados dois qualificadores e três mineradores em uma biblioteca externa. Posteriormente a biblioteca foi adicionada à aplicação *web* e seus mineradores e qualificadores foram registrados nos arquivos de configuração do serviço.

Os três mineradores implementados foram:

- `FindDevelopersByArtifactName`: minera dados sobre uma base de dados de registros previamente extraídos por ferramentas que extraem o histórico de repositórios de códigos fontes. Estes registros são filtrados pelos nomes dos artefatos passados por parâmetro para o minerador.
- `FindDevelopersByWordKey`: também minera dados de uma base extraída do histórico de códigos fontes. Mas utiliza os parâmetros do minerador para identificar mensagens que contenham os termos informados na busca.
- `FindAuthorsByBugInFirebugCSVFile`: minera dados extraídos a partir de arquivo XML gerado pela ferramenta `BUGZILLA` e que contém nomes de desenvolvedores, a lista de bugs e mensagens de abertura.

Estes três mineradores foram adicionadas ao arquivo `webminers.properties` para que fiquem disponíveis durante a execução do sistema. O conteúdo do arquivo é apresentado na figura 17.

```
firebugcsvfile=br.edu.utfpr.ef.miner.implementations.FindAuthorsByBugInFirebugCSVFile
mineralldevelopersbyartifactname=br.edu.utfpr.ef.miner.implementations.FindDevelopersByArtifactName
mineralldevelopersbymessagewordKey=br.edu.utfpr.ef.miner.implementations.FindDevelopersByWordKey
```

Figura 17 - Conteúdo do arquivo webminers.properties após configuração.
Fonte: Autoria própria

Com relação aos qualificadores, foram implementados um qualificador que determina o coeficiente de *expertise* por frequência e outro que determina a *expertise* por atividade mais recente. Seus nomes são `QualifyAuthorByFrequency` e `QualifyAuthorByNewerDate` respectivamente.

Eles operam sobre os dados que são retornados pelos mineradores empregados na busca e quantificam a *expertise* dos participantes a partir destes

dados:

- O qualificador `QualifyAuthorByFrequency` recebe os dados obtidos pelo minerador vinculado à ele. A lista de resultados recebida é percorrida, verificando os desenvolvedores existentes e seus coeficientes de *expertise*. O coeficiente, neste caso, é medido pela frequência de participação de um dado desenvolvedor.
- Já o qualificador `QualifyAuthorByNewerDate` percorre a lista de resultados retornados pelo minerador e gera uma nova lista. Essa nova lista contém um único registro de cada desenvolvedor existente na lista inicial. Cada entrada da nova lista corresponde à participação mais atual do desenvolvedor. Após sua geração, esta nova lista é ordenada por data em ordem decrescente. O coeficiente de *expertise* de cada desenvolvedor é gerado a partir desta lista e é atribuído como sendo o índice desta lista ordenada.

Estes dois qualificadores foram configurados no arquivo `webqualifiers.properties` da aplicação web. O conteúdo do arquivo de configuração é apresentado na figura 18.

```
authorbyfrequency=br.edu.utfpr.ef.qualifier.implementations.QualifyAuthorByFrequency
authorbynewerdate=br.edu.utfpr.ef.qualifier.implementations.QualifyAuthorByNewerDate
```

Figura 18- Conteúdo do arquivo webminers.properties após configuração.
Fonte: Autoria própria

Após a configuração do arquivo `webminers` e do arquivo `webqualifiers`, cinco metodologias que utilizam os três mineradores e os dois qualificadores apresentados foram cadastradas no arquivo `websearchs.xml`. As próximas linhas transcrevem o arquivo XML após as modificações. O conteúdo do arquivo é apresentado na figura 19.

```
<?xml version="1.0" encoding="UTF-8"?>
<searches>
    <search>
        <alias>Authors by frequency in CSV Firebug file</alias>
        <qualifier>authorbyfrequency</qualifier>
        <miner>firebugcsvfile</miner>
    </search>
    <search>
        <alias>MinerALL Authors by frequency and Artifact names</alias>
        <qualifier>authorbyfrequency</qualifier>
        <miner>mineralldevelopersbyartifactname</miner>
    </search>
    <search>
        <alias>MinerALL Authors by newer date and Artifact names</alias>
        <qualifier>authorbynewerdate</qualifier>
        <miner>mineralldevelopersbyartifactname</miner>
    </search>
    <search>
        <alias>MinerALL Authors by frequency and messages wordkey</alias>
        <qualifier>authorbyfrequency</qualifier>
        <miner>mineralldevelopersbymessagewordKey</miner>
    </search>
    <search>
        <alias>MinerALL Authors by newer date and messages wordkey</alias>
        <qualifier>authorbynewerdate</qualifier>
        <miner>mineralldevelopersbymessagewordKey</miner>
    </search>
</searches>
```

Figura 19 - Representação da arquitetura com clientes, serviços, mineradores e qualificadores. Fonte: Autoria própria

A fim de verificar o funcionamento correto das metodologias implementadas foram construídas duas aplicações clientes, apresentadas na próxima seção.

6.2 OS CLIENTES IMPLEMENTADOS

Como dito anteriormente, para verificar o funcionamento do serviço *web* foram construídos dois clientes distintos. O primeiro cliente criado foi um aplicativo

construído utilizando a linguagem JSP (*Java Server Pages*), que consome o serviço de identificação de especialistas e os lista em um navegador *web*. Este aplicativo permite a apresentação gráfica dos resultados em forma de lista contendo os desenvolvedores relacionados.

O segundo cliente criado foi um mensageiro instantâneo que permite a comunicação entre membros utilizando a troca de mensagens síncrona. Este mensageiro executa a busca no serviço, e seus resultados são utilizados para ordenar a lista dos contatos. Para realizar a comunicação síncrona foi utilizado o protocolo de comunicação XMPP (*eXtensible Messaging and Presence Protocol*). Ele foi escolhido por se tratar de um protocolo aberto e amplamente utilizado. Entre os servidores compatíveis com o protocolo XMPP foram testados o Ignite Real Time, OpenFire e o Apache Vysper. O Openfire se mostrou mais simples de configurar e operar que seu concorrente da fundação Apache. A API utilizada para conexão do lado cliente do mensageiro instantâneo será a Ignite Real Time Smack. Tanto o servidor quanto a API cliente da Ignite Real Time serão utilizados tão somente por sua praticidade e facilidade de uso. Nada impede que o mesmo servidor de mensagens instantâneas seja conectado a clientes que utilizem outra API de conexão do lado cliente, ou que seja utilizado outro servidor que suporte XMPP.

Detalhes das implementações dos clientes são apresentadas nas próximas seções.

6.2.1 O CLIENTE WEB

O cliente web consiste de duas páginas JSP (*Java Server Pages*). A primeira página lista as metodologias disponíveis no serviço e permite sua consulta mediante o envio de parâmetros. Essa listagem das metodologias de busca existentes permite que o usuário utilize um mesmo cliente para consultar mais de um tipo de busca por especialistas. E após a escolha do mecanismo de busca a ser utilizado, este é enviado como parâmetro à segunda página juntamente com os parâmetros informados para busca. Neste caso os parâmetros enviados são nomes (ou parte dos nomes) dos artefatos, ou mesmo palavras chaves que restrinjam os valores a serem processados pelos qualificadores. A partir destes valores a segunda página

web requisita a consulta para identificação dos especialistas e apresenta seus resultados. A figura 20 apresenta a primeira página do cliente web.

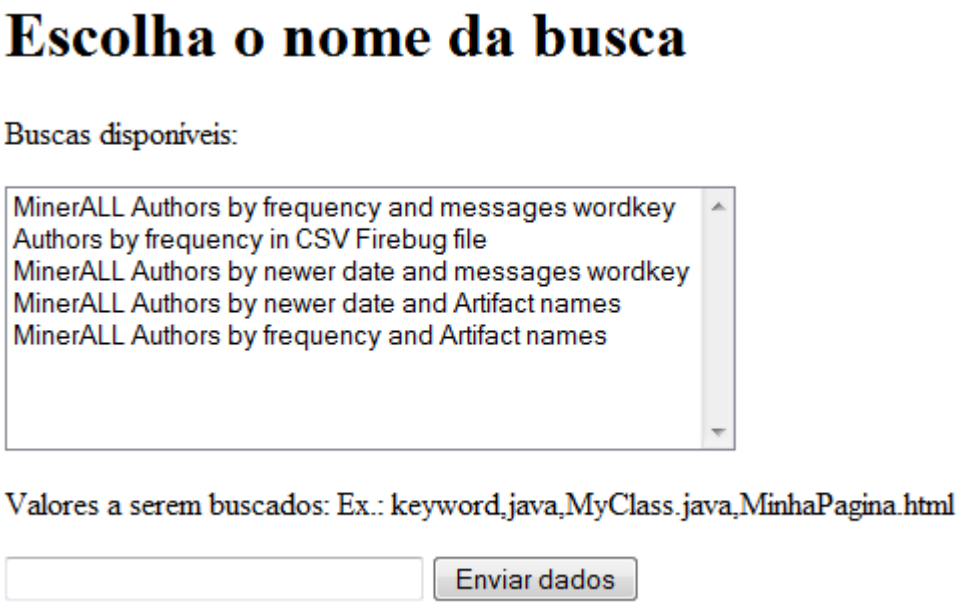

Figura 20 - Página web que permite a seleção envio de termos para busca por *experts*.
Fonte: Autoria própria

A segunda página contém um cliente de serviços *web* vinculado à aplicação servidora que identifica os especialistas. Ela encaminha os parâmetros adquiridos da primeira página por esse cliente de serviço. Após o processamento da requisição, a página obtém uma lista contendo a resposta da requisição do serviço. A partir desta lista é gerada uma lista ordenada pelo coeficiente dos especialistas, que é exibida no navegador como sendo a resposta da requisição realizada. A figura 21 apresenta a página de resultados após a execução de uma busca por especialistas.

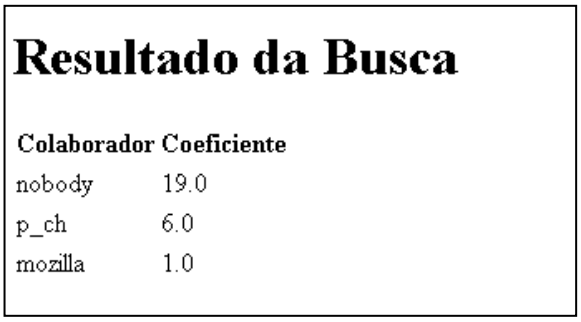

Figura 21 - Página web que exibe os resultados após o processamento da busca por *experts*.

Fonte: Autoria própria

Com este cliente é possível buscar por especialistas utilizando diferentes mecanismos de identificação. Seu resultado é apresentado como uma lista estática. Essa lista contém os nomes e os coeficientes de *expertise* dos desenvolvedores mediante a execução da busca.

6.2.2 O CLIENTE DE INSTANT MESSAGE

O cliente de mensagem instantânea (CMI) possibilita a comunicação síncrona e que os resultados da identificação dos especialistas seja utilizado para ordenar sua lista dos contatos. Para a criação deste cliente foi utilizada a API Ignite Real Time Smack. Ela possibilita a conexão com um servidor de mensagens instantâneas que utilize o protocolo XMPP.

Para que a ordenação seja possível é necessário que os nomes retornados pelo serviço de busca sejam os mesmos dos contatos existentes no CMI. Por isso foi instalado o servidor o Ignite Real Time, OpenFire que possibilita a montagem e configuração de um servidor de mensagens XMPP. Após a configuração do servidor foram adicionados usuários para corresponder aos usuários existentes nos resultados das buscas por especialistas.

Após cada execução de busca, o mensageiro instantâneo irá organizar sua lista de contatos a partir dos pares de valor (desenvolvedor, coeficiente de *expertise*)

obtidos. Os especialistas que forem identificados têm seus nomes destacados com fundo colorido. Essa operação pode ser visualizada na figura 22.

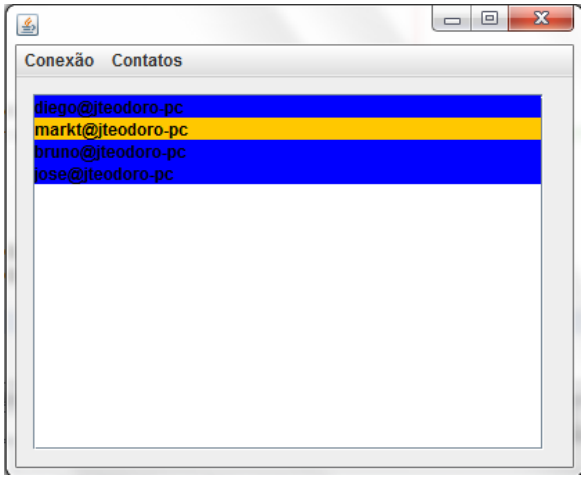

Figura 22 - Lista dos contatos especialistas em destaque.
Fonte: Autoria própria

Ele também possibilita que diferentes metodologias de busca sejam utilizados para identificar os especialistas. A listagem dos mecanismos disponíveis é obtida do próprio serviço de busca e a interface gráfica permite que usuário selecione aquela que melhor se adeque às suas necessidades. A figura 23 apresenta a listagem dos mecanismos de busca retornados pelo serviço web.

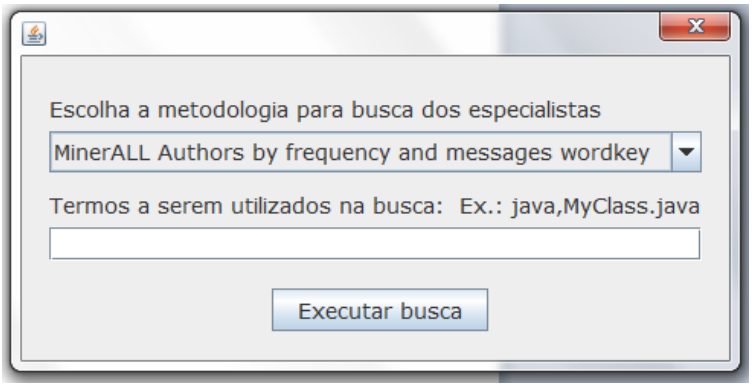

Figura 23 - Caixa de seleção das metodologias de busca dos especialistas.
Fonte: Autoria própria

A partir do mecanismo de busca selecionado é enviada uma requisição ao serviço passando os parâmetros informados na interface gráfica do CMI.

Após a resposta da solicitação, o mensageiro instantâneo realiza a

ordenação da lista de contatos. Com este cliente é possível gerar uma lista de contatos ordenada por coeficiente de *expertise* em um determinado assunto. A partir dessa lista o desenvolvedor pode escolher mais facilmente quem é a pessoa ideal para começar uma interação sobre o assunto em questão.

6.2.3 Compartilhamento das metodologias de identificação

O cliente web e o cliente de mensagens instantâneas compartilham as mesmas metodologias de identificação de especialistas. Ambos adquirem a lista de metodologias disponíveis no serviço e executam-nas mediante a passagem de parâmetros. A funcionalidade de seleção da metodologia de busca possibilita que novos métodos seja adicionados e utilizados sem a necessidade de alteração nestes clientes.

Uma vez que diferentes clientes executam uma mesma busca, eles receberão o mesmo resultado. Cabe então a esses clientes apresentar esses resultados da maneira mais adequada. Desse modo é possível listar os especialistas na página web e ter estes mesmos especialistas ordenados por importância na lista de contatos do mensageiro instantâneo.

7 CONCLUSÃO

O desenvolvimento distribuído de software permite que as organizações aproveitem capital humano distribuído por todo o mundo. Entretanto a comunicação que é essencial para o desenvolvimento é prejudicada pela imposição da distância geográfica. Visto que os desenvolvedores necessitam de informações precisas para realizar seu trabalho e que o compartilhamento eficiente de informações é um fator crítico durante o desenvolvimento de software.

Na tentativa de contornar essas dificuldades foram criadas ferramentas que auxiliam na troca de informações para torná-la mais eficiente. Elas identificam quem pode oferecer ajuda quando um desenvolvedor encontra problemas. E fornecem meios de comunicação uma vez que o especialista seja identificado.

Para verificar essas necessidades dos desenvolvedores de software distribuído foi realizada uma pesquisa de campo. Nela podem-se identificar quatro características para ferramentas para identificação de especialistas em desenvolvimento de softwares, a saber:

- **R1.** Não existe senso comum sobre a melhor fonte de pesquisa para identificar um especialista. Isso depende das necessidades do desenvolvedor, dos recursos disponíveis e do contexto do projeto;
- **R2.** Existem diversos métodos (algoritmos) para ranquear especialistas. Isso inclui diferentes fontes de dados;
- **R3.** Alguns desenvolvedores se sentem mais confortáveis do que outros quando precisam trocar informações com um indivíduo que ele não conhece;
- **R4.** Membros de times globais estão localizados em diferentes países. Isso resulta em diferentes culturas e fuso-horários. Então o cliente que exibe os resultados da busca precisa se adaptar a essa característica.

Baseado nessas características a arquitetura apresentada neste trabalho foi projetada e desenvolvida. Esta ferramenta é uma aplicação web que possibilita a identificação dos especialistas utilizando diferentes metodologias de qualificação (**R2**) que façam uso de dados minerados de diferentes fontes (**R1**). Para possibilitar

a extensibilidade da aplicação foi utilizada uma arquitetura baseada em interfaces e em padrões de projeto, aliada a arquivos de configuração para determinar as metodologias em funcionamento.

A arquitetura da aplicação foi desenvolvida na forma de serviço web, possibilitando que os clientes permaneçam desacoplados da implementação existente no servidor. Isso permite o desenvolvimento de diferentes clientes para apresentar os resultados da busca, possibilitando a adequação às necessidades e características de cada usuário. Com isso pode-se tratar o problema das diferenças pessoais, culturais e temporais existentes no time, uma vez que podem ser criados clientes específicos para atender às necessidades e características de cada grupo de desenvolvedores (**R3** e **R4**). O quadro 1 resume o comparativo entre as ferramentas existentes e o trabalho aqui realizado.

Ferramenta	Atende R1	Atende R2	Atende R3	Atende R4
ExB	✓	✓		
SmallBlue	✓			
Codebook			✓	✓
Presley	✓			
EEL			✓	✓
Proposta deste trabalho	✓	✓	✓	✓

QUADRO 1 – comparativo entre as ferramentas existentes e este trabalho e quais dos requisitos identificados eles atendem.
Fonte: Autoria própria

A extensibilidade da aplicação foi verificada por meio da implementação e configuração de diferentes metodologias de buscas (qualificadores) utilizando diferentes fontes de dados (mineradores). O uso destas metodologias e o desacoplamento dos clientes foi verificado pela implementação de dois clientes que utilizam a aplicação servidora: uma página *web* que permite que sejam listados os especialistas mediante parâmetros; e um cliente de mensagens instantâneas que possibilita a ordenação dos contatos através da busca realizada.

Como trabalhos futuros pretende-se investigar mecanismos de similaridade de código e recomendação para fornecer novos métodos para identificação de especialistas. Além disso, é possível ainda, a criação de *plugins* para IDEs (*Integrated Development Environment*) que possibilitem a consulta por

especialistas a partir da obtenção de informações contextuais da área de trabalho do desenvolvedor.

REFERÊNCIAS

BEGEL, Andrew; DELINE, Robert; ZIMMERMANN, Thomas. Social Media for Software Engineering. In: THE FSE/SDP WORKSHOP ON THE FUTURE OF SOFTWARE ENGINEERING RESEARCH. **Proceedings**... Santa Fe, New Mexico, USA: [s.n.]. 2010.

BLOCH, Joshua. **Effective Java**. 2nd Edition. ed. [S.l.]: Prentice Hall, 2008.

CARTER, Jason; DEWAN, Prasun. Are you having difficulty? In: THE ACM CONFERENCE ON COMPUTER SUPPORTED COOPERATIVE WORK. **Proceedings**... New York: ACM. 2010. p. 211-214.

CATALDO, Marcelo; HERBSLEB, James D.; CARLEY, Kathleen M. Socio-Technical Congruencea framework for assessing the impact of technical and work dependencies on software development productivity. In: THE ACM-IEEE INTERNATIONAL SYMPOSIUM ON EMPIRICAL SOFTWARE ENGINEERING AND MEASUREMENT. **Proceedings**... New York: [s.n.]. 2008. p. 2-11.

DULLEMOND, Kevin; VAN GAMEREN, Ben; VAN SOLINGEN, Rini. Virtual open conversation spacesTowards improved awareness in a GSE Setting. In: IEEE International Conference on Global Software Engineering. **Proceedings**... Princeton: IEEE. 2010. p. 247-256.

EHRLICH, Kate; CHANG, Klarissa. Leveraging expertise in global software teamsGoing outside boundaries. In: THE IEEE INTERNATIONAL CONFERENCE ON GLOBAL SOFTWARE ENGINEERING. **Proceedings**... Washington, DC: IEEE. 2006. p. 149-158.

GOMES, Daniel A. **Web services SOAP em Java**. São Paulo, Brasil: Novatec Editora, 2009.

JOHNSON, Bruce; JOHNSON, Cindy; WOOLFOLK, Walter W.; MILLER, Robert. **Flexible Software Design: Systems Development for Changing Requirements**. [S.l.]: Auerbach Publications, 2005.

JOSUTTIS, Nicolai M. **SOA in Practice**. [S.l.]: O'Reilly Media, 2007.

KUCHANA, Partha. **Software architecture design patterns in Java**. [S.l.]: Auerbach, 2004.

KWAN, Irwin; DAMIAN, Daniela; STOREY, Margaret A. Visualizing a requirements-centred social network to maintain awareness within development teams. In: THE INTERNATIONAL WORKSHOP ON REQUIREMENTS ENGINEERING VISUALIZATION. **Proceedings**... Washington, DC: [s.n.]. 2006. p. 7.

MINTO, Shawn; MURPHY, Gail C. Recommending Emergent Teams. In: THE INTERNATIONAL WORKSHOP ON MINING SOFTWARE REPOSITORIES. **Proceedings**... Washington, DC: [s.n.]. 2010. p. 5.

MOCKUS, Audris; HERBSLEB, James D. Expertise Browsera quantitative approach to identifying expertise. In: THE INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING. **Proceedings**... New York: [s.n.]. 2002. p. 503– 512.

MORAES, Alan K.; SILVA, Eduardo, T. C. C.; BARBOSA, Yuri; MEIRA, Silvio. Recommending Experts Using Communication History. In: THE INTERNATIONAL WORKSHOP ON RECOMMENDATION SYSTEMS FOR SOFTWARE ENGINEERING. **Proceedings**... New York: [s.n.]. 2010. p. 41-45.

ROBINSON, Marcia; KALAKOTA, Ravi. **Offshore Outsourcing: Business Models, ROI and Best Practices**. Alpharetta, GA: Mivar Press, 2004.

ROY, Jaideep; RAMANUJAN, Anupama. **XML: data's universal language**. IT Professional, 2, May 2000. 32-36.

SAINT-ANDRE, Peter. **XMPP: lessons learned from ten years of XML messaging**. Communications Magazine, April 2009. 92-96.

SHAMI, N.Sadat; EHRLICH, Kate; MILLEN, David R. Pick Me! Link selection in expertise search results. In: THE ANNUAL SIGCHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS. **Proceedings**... New York: [s.n.]. 2006.

TRINDADE, Cleiton C.; MORAES, Alan K. O.; MEIRA, Silvio L. Comunicação em equipes distribuídas de desenvolvimento de SoftwareRevisão sistemática. In: THE EXPERIMENTAL SOFTWARE ENGINEERING LATIN AMERICAN WORKSHOP. **Proceedings**... Salvador, Brasil: [s.n.]. 2008.

TRINDADE, Cleiton C.; MORAES, Alan K.; BARBOSA, Yuri. Presley: uma ferramenta de recomendação de especialistas no contexto de código-fonte para apoio à colaboração em desenvolvimento distribuído de software. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA SOFTWARE. **Anais**... Fortaleza, Ceará, Brasil: [s.n.]. 2009.

WALLS, Craig; BREIDENBACH, Ryan. **Spring in Action**. 2nd Edition. ed. [S.l.]: Manning, 2007.

APÊNDICE A: Pesquisa de campo

Esta seção apresenta o questionário apresentado aos desenvolvedores de equipes distribuídas. A pesquisa foi realizada nos meses de março e abril de 2011 e envolveu apenas membros de projetos distribuídos. Os indivíduos que participaram da pesquisa possuem experiência entre três e trinta anos como desenvolvedores e de um há dez anos com desenvolvimento distribuído.

Questionário apresentado na pesquisa de campo

Survey on collaboration in distributed software development teams

This survey is part of a academic research that aims to provide ways to support collaboration among distributed team members. The data collected on this survey will be used to build collaboration tools according to the needs raised by it.

Thanks for helping our research by answering this survey. The average time to complete the survey is around 20 minutes.

The survey is composed of two parts:

The first part is related to your profile and your team

The second part is related to the ways you look for information/support while developing

There are 25 questions in this survey

Profile

This set of question is related to your personal/technical profile

[1] What's your age?

[2]How many years of experience on software development do you have? *

[3] How many years have you been working on distributed teams? *

[4] I develop for: *

Please choose **all** that apply:

- A company (proprietary/industrial software)
- Free/Open Source Software Project

[5] In which city/country do you work? *

[6] In which other cities/countries are your teammates?

Just answer this question in case you know where your teammates are. In case you don't know their location, answer: DO NOT KNOW

[7]What is the size of your local team? *

Please choose **only one** of the following:

- I don´t have a local team, I work alone

- From 2 to 5 members
- From 6 to 10 members
- From 11 to 20 members
- More than 20 members

[8]How many of your team members work geographically distant from you? *
Please choose **only one** of the following:

- I don´t know / I work on Open Source Projects
- Less than 5 members
- From 6 to 10 members
- From 11 to 20 members
- More than 20 members

[9] What is your technical skill level (self-evaluation)? *
Please choose **only one** of the following:

- Basic
- Intermediate
- Advanced
- Expert

[10] What's your role/position? *
Please choose **only one** of the following:

- Developer
- Business Analyst
- Architect
- Requirements Analyst
- Tester
- Team Leader/Manager
- Other

Collaboration, coding and experts finding

[11] When working on a specific software artifact (class, method, library, etc..) what information do you consider important to facilitate / support your work?
*Please choose the importance grade you give to the item (0 means not important at all; 4 means very important) **

Please choose the appropriate response for each item:

	1	2	3	4	5
Which developer has developed the artifact					
Which developer is the expert on this artifact					
Which other artifacts depends on this artifact					
From which artifacts does this artifact depends on					
When was the last change of this artifact					
Which bugs related to this artifact have already being reported					
What are the email/forum threads related to this artifact					
To which artifacts is this artifact related to					
Which teams have worked on this artifact					

Please choose the importance grade you give to the item (1 means not important at all; 5 means very important)

[12]If there are some other information that you consider important and it is not listed above, please list them

[13]When you have difficulty on understanding a particular piece of code, which of the following do you consider the most direct way to solve your problem? (Please select at most 2 items) *
Please choose **all** that apply:

- Search for solutions in forums / emails threads (maintained by the project team)
- Talk to people close to you looking for someone who has worked on that project
- Try to find out who are the people with more experience on that piece of code and contact them
- Solve the problem by yourself, without consulting any other sources
- Other:

[14]Suppose that you need to contact the owner of a piece of code and do not know who he/she is. How would you find this person?

[15]Suppose you need to solve any problem/issue in your code. To do so you need to contact an unknown person. Do you feel comfortable with starting this conversation? *
Please choose **only one** of the following:

- Yes
- No

[16]What is(are) the reason(s)?
Only answer this question if the following conditions are met:
° Answer was 'No' at question '15 [4]' (Suppose you need to solve any problem/issue in your code. To do so you need to contact an unknown person. Do you feel comfortable with starting this conversation?)

[17]Have you ever faced a problem when your code crashes due to simultaneous change of any library / code related? *
Please choose **only one** of the following:

- Yes
- No

[18]How have you solved the problem?
Only answer this question if the following conditions are met:
Answer was 'Yes' at question '17 [5]' (Have you ever faced a problem when your code crashes due to simultaneous change of any library / code related?)

[19]Do you consider that the developers who made more commits / checkins for a particular artifact in the SCM can be considered experts of that artifact?
SCM: Software Configuration Management (CVS, MKS, ClearCase, SVN, GIT etc.)*
Please choose **only one** of the following:

- Yes
- No

[20] Why?
Only answer this question if the following conditions are met:
Answer was 'No' at question '19 [6]' (Do you consider that the developers who made more commits / checkins for a particular artifact in the SCM can be considered experts of that artifact?SCM: Software Configuration Management (CVS, MKS, ClearCase, SVN, GIT etc.))

[21] Do you consider that the most recent commits are more relevant than the old ones?
Only answer this question if the following conditions are met:
Answer was 'Yes' at question '19 [6]' (Do you consider that the developers who made more commits / checkins for a particular artifact in the SCM can be considered experts of that artifact?SCM: Software Configuration Management (CVS, MKS, ClearCase, SVN, GIT etc.))

[22] Have you ever used this mechanism to find an expert in a particular artifact? Please make a comment
Only answer this question if the following conditions are met:
° Answer was 'Yes' at question '19 [6]' (Do you consider that the developers who made more commits / checkins for a particular artifact in the SCM can be considered experts of that artifact?SCM: Software Configuration Management (CVS, MKS,

ClearCase, SVN, GIT etc.))

[23]Have you ever searched for experts on a particular piece of code / functionality on mailing lists, e-mail threads, bugtrackers? *
Please choose **only one** of the following:

- Yes
- No

[24]Do you consider a good approach searching for experts on a particular piece of code / functionality on mailing lists, e-mail threads, bugtrackers? *
Please choose **only one** of the following:

- Yes
- No

[25]Do you think it is more effective to search for experts in textual materials (mailing lists, emails, newsgroups) or in software configuration management systems? *
Please choose **only one** of the following:

- Software configuration management (SCM) systems
- Textual materials (mailing lists, emails, newsgroups)
- Both of them are effective
- None of them
- Thanks a lot. We appreciate your help and the time spent on our survey.

APÊNDICE B: Diagrama de classe da arquitetura

Esta seção apresenta o digrama de classes completo da arquitetura.