An Expert Recommender System to Distributed Software Development: Requirements, Project and Preliminary Results

Um Sistema de Recomendação de Especialistas em Desenvolvimento Distribuído de Software: Requisitos, Projeto e Resultados Preliminares

Cleyton C. da Trindade
Centro de Informática
Universidade Federal de Pernambuco
Recife, PE, Brasil
cct@cin.ufpe.br

Yuri A. M. Barbosa
Universidade Federal de Pernambuco
Recife, PE, Brasil
yamb@cin.ufpe.br

Alan K. O. Moraes
Departamento de Informática
Universidade Federal da Paraíba
João Pessoa, PB, Brasil
alan@di.ufpb.br

Jones O. Albuquerque
Departamento de Estatística e Informática
Universidade Federal Rural de Pernambuco
Recife-PE, Brasil
joa@deinfo.ufrpe.br

Silvio R. L. Meira
Centro de Informática
Universidade Federal de Pernambuco
Recife, PE, Brasil
srlm@cin.ufpe.br

***Abstract* — In distributed software development, geographical and temporal distances turn communication inefficient and affect the various team's perceptions levels – proximity, activity, awareness, process and perspective. The low level of perception turns the task of expert location hard and not being able to quickly locate the experts in the source code during the implementation makes the project slower, consequently delaying its schedule. This paper proposes Presley, a tool to reduce these delays through identification and recommendation of experts in some areas of the source code using the project communication, decreasing the waiting time for help. We also present preliminary results which show the practical potential of our approach**

Keywords-Distributed Software Development; Expert Recommender System; Knowledge Management

***Resumo* — No desenvolvimento distribuído de software, as distâncias geográfica e temporal geram deficiências na comunicação e nos diversos níveis de percepção – proximidade, atividade, presença, processo e perspectiva – da equipe. O enfraquecimento das percepções dificulta o reconhecimento rápido das pessoas capacitadas a ajudar outros desenvolvedores com problemas no momento da codificação, tornando a execução do projeto mais lenta, conseqüentemente provocando atrasos no cronograma. A ferramenta Presley objetiva diminuir estes atrasos através da identificação e recomendação dos desenvolvedores especialistas em determinadas porções do código-fonte, reduzindo o tempo de busca e de espera por ajuda. O trabalho apresenta resultados preliminares que demonstram o potencial prático da ferramenta**

Palavras-chave: Desenvolvimento Distribuído de Software, Sistema de Recomendação de Especialista, Gerenciamento do Conhecimento

INTRODUÇÃO

Várias empresas de software optaram por formar equipes de desenvolvimento com seus integrantes distribuídos em diferentes localizações, chegando muitas vezes a ultrapassar as barreiras continentais, para suprir a necessidade de conseguir os profissionais mais especializados, reduzir o custo de desenvolvimento, ter presença globalizada e alcançar maior proximidade com os clientes [3].

O êxito na construção de software envolve competência em três domínios: Engenharia de Software, Problema e Aplicação. O Domínio da Engenharia de Software compreende o conhecimento de técnicas, métodos, processos e ferramentas para a construção de software economicamente viável, confiável e eficiente. Este domínio diz como devemos construir o software. O Domínio do Problema diz respeito ao entendimento de requisitos, regras de negócio, necessidades e desejos das pessoas envolvidas no projeto. Ele informa o que deve ser construído. Por fim, o Domínio da Aplicação consiste de todo conhecimento necessário para compreender como o Domínio do Problema está sendo transformado em software, isto é, transcrito e codificado nos

artefatos e código-fonte do projeto. O último domínio reflete as características e propriedades do software em construção.

Porém, se o desenvolvimento de software tradicional já é reconhecidamente uma atividade complexa quando realizada da forma tradicional [4; 2; 6], a complexidade é ainda maior e elevada a níveis mais difíceis de serem tratados em desenvolvimento distribuído [1]. Ao observar especificamente os processos de comunicação em equipes distribuídas, nota-se a falta de oportunidades para interações síncronas, que ocorrem de maneira espontânea e informal no desenvolvimento tradicional, dificultando o compartilhamento do conhecimento ganho sobre o software em si que está sendo escrito.

Este problema é mais claramente percebido quando é necessário obter ajuda de outro membro remoto do time, porque o acesso às pessoas de times remotos e as opções de comunicação síncrona são restritas. As limitações tornam lento o inicio do processo de colaboração entre os integrantes da equipe e dificultam a descoberta dos especialistas em assuntos específicos da implementação do projeto. Como conseqüência, projetos distribuídos têm custos adicionais com atrasos e retrabalhos [7].

A ferramenta Presley busca diminuir os atrasos gerados na comunicação de equipes distribuídas ao identificar e recomendar os especialistas existentes em um projeto àquelas pessoas que buscam por ajuda durante suas atividades de codificação do projeto, reduzindo o tempo de espera e evitando desperdício de esforço.

O restante deste trabalho está organizado da seguinte forma: A Seção 2 destaca as dificuldades enfrentadas pelos times distribuídos para estabelecer uma boa colaboração; trabalhos relacionados são discutidos na Seção 3; a Seção 4 introduz a ferramenta Presley, sua arquitetura e detalha o mecanismo de recomendação; a Seção 5 apresenta os resultados preliminares em um grande projeto distribuído; por fim, a Seção 6 apresenta as considerações finais.

Dificuldades na Colaboração em Desenvolvimento Distribuído de Software

Equipes co-localizadas gastam boa parte do seu tempo em comunicação, em média, consomem 75 minutos do seu dia em interações não programadas [20]. Contudo as distâncias geográfica e temporal diminuem a freqüência de comunicação entre as equipes, o que interfere na forma de interagir das pessoas [10].

A comunicação age como um fundamental componente entre as práticas e processos do desenvolvimento de software e ganha forte atenção no Desenvolvimento Distribuído de Software (DDS) por causa do seu impacto na velocidade de execução das atividades [16]. Embora a quantidade de eventos que atrasam os cronogramas seja semelhante entre projetos tradicionais e distribuídos, um time distribuído é duas vezes e meio mais lento na resolução destes problemas, uma diferença significativa [10], embora o resultado seja contestado por Nguyen, Wolf e Damian [18].

Executou-se uma Revisão Sistemática da Literatura (RSL) [15] para entender os conceitos-chave e as dificuldades freqüentemente envolvidos na comunicação de projetos distribuídos, bem como as experiências nesta área relativas a processos, técnicas e ferramentas utilizadas para minimizar este problema [22]. Essa revisão também contribuiu na definição de requisitos de bons meios de comunicação em equipes distribuídas, que possibilitam interações mais eficientes, precisas e com menor custo para o projeto.

Entre os pontos destacados pela RSL, a falta de percepção mostrou-se um fator de suma preocupação no DDS, pois está presente em diversos aspectos na comunicação entre times. Foi necessário classificar as diferentes definições de percepção encontradas na literatura em cinco dimensões, a saber: atividade, proximidade, presença, processo e perspectiva. Analisar o problema da comunicação sob a ótica das diferentes dimensões da percepção ajuda-nos a compreendê-lo melhor. As dimensões de percepção são enumeradas a seguir:

Percepção de Atividade (O que eles estão fazendo?): É a importância de conhecer quem

fez o que e quando, porque sem essa propriedade é difícil discernir quem tem conhecimento sobre determinados assuntos para esclarecer dúvidas, discutir melhores soluções e repassar conhecimento [19; 8; 10]. Em DDS, é menos aparente quem são e o que fazem as pessoas em times remotos.

Percepção de Proximidade (Com quem minhas atividades relacionam-se?): É a necessidade que os indivíduos possuem de saber quem está próximo a eles (em termos de estrutura) e sobre as dependências do sistema em desenvolvimento (questões de escopo e efeito causado por mudanças), por exemplo, quantas e quais pessoas serão afetadas caso um determinado componente seja alterado [8]. As distâncias física e de fusos-horário reduzem a percepção de proximidade.

Percepção de Presença (Quando posso contatá-lo?): É a possibilidade de saber qual o melhor momento de iniciar um contato com pessoas de outro time sem que interrompam suas atividades em momentos inapropriados [8; 10]. A percepção de presença é enfraquecida devido à predominância da comunicação assíncrona.

Percepção de Processo (onde estamos no projeto?): É a capacidade de entender completamente quais foram os principais marcos de entrega, os requisitos das tarefas de seus participantes remotos e como eles impactam na sua própria tarefa [13]. Devido às peculiaridades das institucionais locais, os participantes das equipes têm dificuldades em compreender como outros times estão se desenvolvendo.

Percepção de Perspectiva (o que eles estão pensando e por quê?): É a compreensão e entendimento de como as outras partes vêem o mesmo projeto, a cognição coletiva [13]. Nos projetos distribuídos, está relacionada aos questionamentos pelos membros das equipes [13] do porquê de seus colegas remotos falharem ao seguir uma sugestão ou o que eles pensarão sobre alguma contribuição.

Um dos aspectos mais prejudicados com o enfraquecimento das várias percepções – evidente nos problemas das percepções de atividades, proximidade e presença – está na identificação de especialistas em determinadas partes do código-fonte, no contexto de implementação, tornando vagarosa e não muito eficiente a escolha das pessoas necessárias [11]. Tais pessoas (os especialistas) seriam capazes de ajudar os demais membros do time quando surgem dúvidas na fase de construção do software. Como as equipes podem ter um tempo de sobreposição de horário de trabalho muito curto, a rápida identificação da pessoa mais provável a responder mensagens de dúvidas é, potencialmente, uma grande oportunidade para reduzir os atrasos gerados na comunicação, principalmente, assíncrona entre equipes distribuídas.

Presley, a ferramenta descrita na sequência, busca diminuir as deficiências na identificação de especialistas, recomendando-os através de uma máquina de inferência. Desta forma, provê meios que ajudam os membros das equipes encontrarem as pessoas mais qualificadas para solucionar suas dificuldades de implementação.

Trabalhos Relacionados

O STeP_IN [24] tem como objetivo recomendar um conjunto de especialistas em um determinado assunto do projeto. Essa lista é formada a partir do código-fonte e da comunicação realizada durante o desenvolvimento do projeto. Com essas informações o sistema cria uma rede com os relacionamentos mostrados na Figura 1, que possibilita encontrar os especialistas em cada artefato do software melhor relacionado com o remetente do e-mail.

O Expert Finder [21] é uma ferramenta com o objetivo de capturar o conhecimento organizacional através da descoberta dos especialistas. Ela toma como base os documentos gerados pelos membros do projeto. Através do uso de algoritmos de modelagem textual, baseado na filtragem vetorial, esses documentos são analisados e são os selecionados aqueles que atendam a algum dos filtros disponíveis. O primeiro filtro indica que o participante deve está ligado a maior quantidade de documentos encontrados, enquanto o segundo indica que os termos presentes nos documentos selecionados devem recebem pesos diferentes, dependendo da sua localização no texto (titulo, resumo etc.), para indicar aqueles com maior envolvimento com o assunto.

O Expertise Recommender [17] usa artefatos de software para gerar recomendações para um dado problema. O sistema utiliza os registros de experiências entre as pessoas e o histórico dos artefatos para identificar as pessoas com o conhecimento desejado. A recomendação é feita após filtragem dos candidatos por critérios organizacionais, sociais e de preferência do solicitante.

O Theseus [5] objetiva mostrar ao gerente do projeto a dissonância entre a rede de comunicação dos membros de um time e a rede de interdependências do código-fonte de seus respectivos autores. A ferramenta utiliza emails e histórico do código-fonte como entradas para a geração destas duas redes. Embora ela não faça explicitamente nenhuma recomendação, a visualização provida é bastante interessante como uma ferramenta analítica.

Presley tem um foco bastante específico: recomendar especialistas no Domínio da Aplicação. A ferramenta apóia-se nas informações do próprio código-fonte, relacionamentos de suas entidades codificadas, histórico do controle de versão e registros de comunicação entre os membros dos times. Os resultados obtidos são bastante promissores.

PRESLEY – UM SISTEMA DE RECOMENDAÇÃO DE ESPECIALISTAS EM DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE

No espaço de duração do desenvolvimento do projeto, seus participantes compartilham conhecimentos sobre vários assuntos necessários e relacionados à condução das suas atividades e à interação com outros desenvolvedores [8; 19; 12].

Contudo, as dificuldades encontradas em ambiente distribuído, enumeradas anteriormente, não ajudam a compartilhar esse tipo de conhecimento. A falta de conhecimento profundo e seguro da aplicação em si, isto é, do código-fonte que está sendo escrito, tem o potencial de afetar negativamente a produtividade das equipes, principalmente em DDS. A proposta de Presley é identificar e recomendar os especialistas em determinadas áreas do código-fonte do projeto para melhorar a colaboração entre os desenvolvedores. A recomendação de especialistas no contexto da implementação supre uma importante parte da percepção e do compartilhamento de conhecimento necessários numa comunicação eficiente.

Presley utiliza o framework descrito por Ye e colegas [24], que realiza união entre elementos de conhecimento (os desenvolvedores, os documentos e os códigos–fonte) envolvidos na realização do projeto para obter o ecossistema entre esses e selecionar os especialistas que possam responder aos problemas enfrentados pelos desenvolvedores na fase de implementação. Os nós da rede (representados por código-fonte, documento e desenvolvedor) mostram o relacionamento entre as pessoas e as informações geradas durante o projeto, como apresentado na Figura 1.

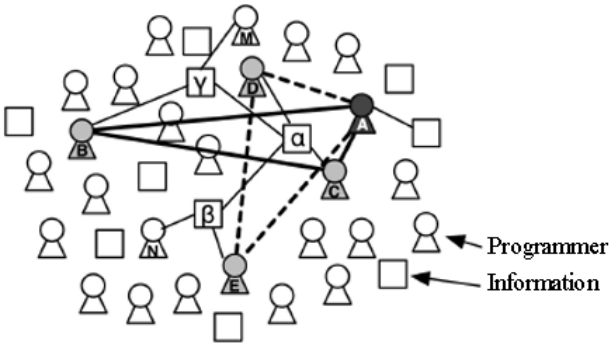

Figure 1. Rede de atores de um projeto de software [24]

Além do código-fonte e dos documentos, Presley também faz uso da comunicação ocorrida no projeto entre os desenvolvedores através de listas de discussão e mensagens na própria ferramenta. Diferentemente de Ye e colegas, que utiliza a comunicação do projeto apenas como meio de criação da rede de contatos de cada desenvolvedor, Presley considera o conteúdo das mensagens para identificar o conhecimento abordado em cada uma e, assim, recomendar as pessoas que mais sabem sobre ele. Desta forma, o Presley recebe como parâmetros o código-fonte, os artefatos e registros de comunicação entre pares no projeto. A partir destas entradas são criados os seguintes relacionamentos:

- Documento–Código-fonte: Formados pela análise entre as palavras-chave de documentos e os comentários em códigos-fonte;
- Desenvolvedor–Código-fonte: Formados pelos registros encontrados na ferramenta de controle de versão;

- Código-Fonte–Código-fonte: Formados pelas dependências entre códigos-fonte;
- Desenvolvedor–Desenvolvedor: Formados através da comunicação realizada em listas de discussões e na ferramenta.

Os parâmetros escolhidos trazem indícios dos detentores de conhecimento envolvidos com esses elementos desde sua concepção. A relação entre a execução das atividades do projeto e seus respectivos artefatos fornecem ao Presley as informações necessárias para inferir quem é o especialista em cada parte (pacote, classe, método) do software, juntamente com a interação ocorrida entre os indivíduos. Tal interação permite o sistema identificar quais são seus interesses dentro do projeto, contribuindo na realização da inferência, porque as pessoas tendem a escrever sobre tópicos do projeto que dominam e dado que é possível identificar qual conhecimento é abordado em cada email.

Arquitetura

Para proporcionar a integração com a IDE Eclipse, o Presley foi desenvolvido sobre a arquitetura cliente-servidor. A Figura 2 ilustra a arquitetura da ferramenta e seus principais componentes.

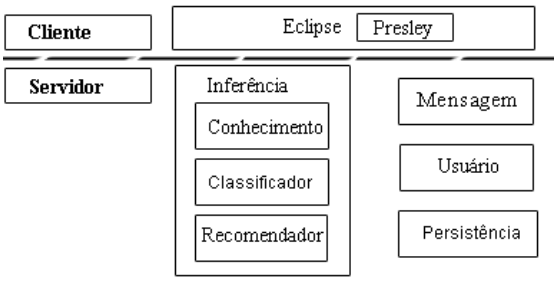

Figure 2. Framework Arquitetural da Ferramenta

O cliente, desenvolvido como um plug-in do Eclipse (Figura 3), é responsável por coletar e enviar informações fornecidas pelos usuários para o servidor, através de Java RMI. O processo de identificação dos especialistas inicia-se quando um desenvolvedor envia uma mensagem de dúvida nessa camada. O usuário, ao escrever uma mensagem, informa o código-fonte relacionado a sua questão. Após este passo, o cliente encarrega-se de buscar os arquivos que fazem referência a esse código, antes de enviá-los ao servidor.

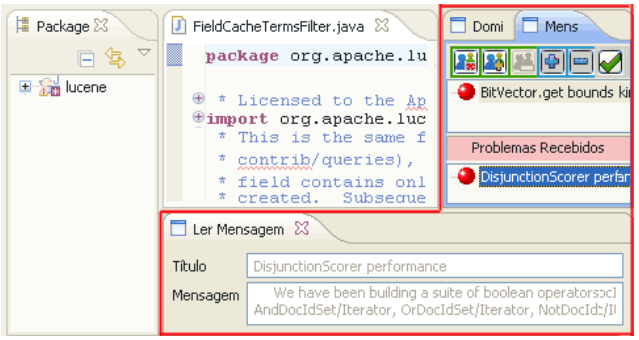

Figure 3. Presley em destaque na IDE Eclipse

Já no lado servidor, existe uma divisão de papeis entre os seguintes componentes: Usuário, Mensagem, Inferência e Persistência.

Componente Usuário é responsável por administrar as pessoas envolvidas no projeto e usuárias da ferramenta. Dentre as informações de identificação do usuário, esse serviço solicita seu grau de conhecimento prévio sobre os tópicos do Domínio do Problema.

Componente Mensagem permite a troca de mensagens entre os desenvolvedores e tem a responsabilidade de anexar ao seu conteúdo os comentários dos arquivos fornecidos pelo cliente. Este componente também permite selecionar as contribuições consideradas como úteis na solução de seus problemas e encaminhar esta informação ao componente Inferência, melhorando os resultados da ferramenta.

Inferência divide-se em três sub-componentes: Classificador, Conhecimento e Recomendador). Antes de realizar recomendações, é necessário cadastrar todos os conhecimentos envolvidos no projeto e associar cada artefato do projeto a um deles. O componente Classificador identifica as palavras-chaves de um documento ou mensagem. O componente calcula a quantidade de palavras utilizadas e retira do texto as stopwords e os caracteres inválidos. A lista de stopwords contém artigos, advérbios, conjunções, preposições, pronomes e interjeições [23]. Após a identificação das palavras-chave, o componente Classificador calcula a freqüência das palavras no texto. Enquanto que o componente Conhecimento recebe uma mensagem, analisa-a e, a partir de suas palavras-chave, determina o seu conhecimento dentre os já previamente cadastrados. Já o componente de Recomendador identifica as pessoas mais

qualificadas para responderem a mensagem.

Por fim, o componente Persistência é responsável por armazenar as informações sobre as mensagens trocadas, os conhecimentos envolvidos no projeto, os códigos-fonte citados e o histórico dos usuários no banco de dados.

Inferência

Para classificar as mensagens recebidas, uma técnica de similaridade difusa foi adaptada para determinar o grau de semelhança da lista de palavras-chave de um documento ou mensagem com a lista de palavras-chave de cada conhecimento. Para cada termo comum entre as listas, calcula-se o grau de igualdade de seus escores de relevância através da função apresentada na Figura 4. Para realizar o cálculo do grau de igualdade, a fórmula recebe o escore de relevância da palavra-chave de cada lista pelas variáveis a e b. O valor resultante c deve estar no intervalo [0,1]; caso isso não aconteça, realiza-se uma normalização. A normalização é feita assumindo o valor 1 quando o resultado é maior que 1 e 0 quando menor que 0.

$$gi(a,b) = \frac{1}{2}\left[(a \to b) \wedge (b \to a) + (\neg a \to \neg b) \wedge (\neg b \to \neg a)\right]$$

Onde:

$a \to b = \max\{c \in [0,1] \mid a \times c \leq b\}$[1];

$\neg a, \neg b = 1 - a$ *ou* $1 - b$;

$\wedge = min$.

Figure 4. Fórmula do cálculo do grau de igualdade [9]

Ao término dos cálculos de igualdade entre os termos, calcula-se o grau de similaridade entre o texto e o conhecimento pela fórmula na Figura 5. Esse cálculo é repetido em cada conhecimento existente no banco de dados. Aquele que obtiver o maior grau de similaridade será o definido para a mensagem.

Com a definição do conhecimento abordado na mensagem, o componente Recomendador identifica as pessoas mais qualificadas para responderem a solicitação. Esse processo inicia-se com a busca dos usuários que mais trocaram mensagens referentes ao mesmo conhecimento, encontrado no passo anterior, e que têm melhor relacionamento com o usuário remetente. Em seguida é obtido o nível de participação dos desenvolvedores nos arquivos anexos à mensagem, através do histórico da ferramenta de controle de versões. O grau de conhecimento fornecido no cadastro do usuário é também outro parâmetro deste cálculo. Os desenvolvedores que obtiverem o maior nível de participação, tanto no desenvolvimento das classes quanto nas trocas de mensagens, serão recomendados ao usuário que enviou a dúvida e receberão o problema a ser solucionado.

$$gs(X,Y) = \sum_{h=1}^{k} gih(a,b)$$

Onde:

gs o grau de similaridade entre os documentos X e Y;
gi o grau de igualdade entre os pesos do termo h (peso a no documento X e peso b no documento Y);
h é um índice para os termos comuns aos dois documentos;
k é o número total de termos comuns aos dois documentos.

Figure 5. Fórmula da média por operadores "fuzzy" [9]

Resultados Preliminares

Com o propósito de verificar a eficácia das recomendações de especialistas feitas pelo Presley, foi realizada uma simulação para verificar a acurácia das recomendações realizadas pelo sistema [14]. A acurácia foi calculada como a razão entre o número de casos distintos em que o Presley conseguiu recomendar um especialista com sucesso e o número total de casos de solicitações de ajuda.

Para ter-se uma base de comparação com Ye, escolheu-se o mesmo projeto, o Apache Lucene-Java. O projeto é uma API de indexação e busca de documentos, escrita em Java e desenvolvida como um projeto de código aberto, portanto desenvolvido de forma distribuída e com as dificuldades expostas já discutidas.

A simulação foi produzida tendo como entrada os registros da lista de discussão, o código-fonte e seu respectivo histórico na ferramenta de controle de versão, e a documentação do projeto, disponíveis em http://lucene.apache.org. Ao todo foram utilizados 17.473 emails registrados entre 2004 e 2008, enviados pelos 1013 participantes da lista de discussão. O código-fonte tem mais de 140.000 linhas de código.

A ferramenta recebeu como entrada 20 emails enviados para a lista de discussão do projeto em 2009, escolhidos aleatoriamente, e verificou-se se os especialistas recomendados por Presley foram as mesmas pessoas que realmente responderam ao pedido de ajuda na lista de

discussão do Lucene, os verdadeiros especialistas.

A acurácia das recomendações foi de 80% de acerto, isto é, entre os 20 e-mails enviados, 16 casos mostraram que ao menos um desenvolvedor reconhecido pelo sistema como especialista realmente respondeu a solicitação de ajuda na lista do projeto. Este resultado é superior aos 75% de acurácia do trabalho de Ye e colegas [24].

Solicitante	Assunto do email	Desenvolvedores que responderam					Desenvolvedores recomendados				
D001	problem with indexreader.reopen()	D003					D003	D034	D010	D025	D014
D002	porting benchmark suite	D014	D003	D023	D009		D014	D034	D043	D016	D030
D003	failure in testtrierangequery	D007	D027	D033			D042	D014	D014	D007	D028
D004	sorting lucene search results	D020					D010	D034	D003	D025	D044
D005	bloomfilter-s with lucene	D021	D028	D001			D003	D034	D023	D014	D027
D006	question on lucene search	D014					D010	D014	D034	D026	D003
D004	bubbling up newer records	D022	D029	D003	D039		D010	D014	D034	D026	D003
D007	question on small optimization in trieutils	D023					D003	D023	D010	D034	D026
D008	testindexinput test failures on …	D023	D030	D003			D003	D010	D034	D014	D036
D009	bitvector.get bounds checking	D003	D001	D007			D003	D010	D034	D042	D044
D010	Trierange	D007	D031	D034			D003	D014	D034	D044	D042
D011	sort lucene results	D007	D027				D010	D014	D034	D026	D003
D012	integrating language models into lucene	D024	D026	D014	D040		D003	D010	D014	D044	D023
D013	getting tokens from search results…	D025	D027	D035	D003	D014	D003	D014	D010	D016	D046
D014	empty sink tokenizer	D003	D010				D003	D010	D044	D044	D046
D015	disjunctionscorer performance	D026					D003	D042	D044	D025	D009
D016	filesystem based bitset	D003 D010	D014 D034	D041 D036	D026 D002	D028	D003	D010	D014	D034	D044
D017	indexwriter.rollback() logic	D009	D027	D037	D003		D003	D010	D014	D044	D034
D018	fips compliance?	D003	D032	D038			D003	D010	D044	D034	D014
D019	iterable in fieldcachetermsfilter	D003	D007				D003	D014	D045	D026	D022

CONSIDERAÇÕES FINAIS

As separações geográfica e temporal geram deficiências nos vários níveis de percepção – proximidade, atividades, presença, processo e perspectiva – e na distribuição de conhecimento das atividades executadas durante o desenvolvimento de um projeto. As deficiências são evidenciadas quando surge a necessidade de esclarecimentos, devido à dificuldade em saber quem tem conhecimento sobre o assunto que se deseja tratar e quando será possível contatar esta pessoa, conseqüentemente tornando lento o inicio do processo de colaboração entre os integrantes.

A ferramenta proposta agiliza o processo de descoberta de especialistas em um projeto, no nível de código-fonte, a partir das descrições das dificuldades enfrentadas durante a implementação, do próprio código-fonte e do histórico de comunicação do projeto.

Resultados preliminares, no projeto Apache Lucene-Java, apontam acurácia de 80% nas recomendações, índice elevado e satisfatório, atestando assim a qualidade da ferramenta. Espera-se melhorar estes resultados através da agregação de mais informações de contexto, da modelagem temporal dos especialistas e da experimentação de outros algoritmos de similaridade

REFERÊNCIAS

[1] Ågerfalk, P., Fitzgerald, B., Holmström, H., Lings, B., Lundell, B. and Ó Conchúir, E. (2005) "Framework for considering Opportunities and Threats in Distributed Software Development", In DiSD'05: Proceedings of the International Workshop on Distributed Software Engineering, 47-61, Paris, France

[2] Brooks, F. P. (1978). The Mythical Man-Month: Essays on Softw. 1st. Addison-Wesley Longman Publishing Co., Inc.

[3] Carmel, E. (1999). Global Software Teams: Collaborating Across Borders and Time Zones. Prentice Hall.

[4] Curtis, B., Krasner, H., and Iscoe, N. 1988. A field study of the software design process for large systems. Commun. ACM 31, 11 (Nov. 1988), 1268-1287.

[5] de Souza, C. R., Quirk, S., Trainer, E., and Redmiles, D. F. 2007. Supporting collaborative software development through the visualization of socio-technical dependencies. In GROUP '07: Proceedings of the 2007 international ACM Conference on Supporting, pp 147-156.

[6] Espinosa, A. J. and Carmel, E. 2004. “The Effect of Time Separation on Coordination Costs in Global Software Teams: A Dyad Model”. In Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences, IEEE Computer Society, Washington, DC, USA.

[7] Espinosa, A. J. and Pickering, C. 2006. The Effect of Time Separation on Coordination Processes and Outcomes: A Case Study. In Proceedings of the 39th Annual Hawaii International Conference on System Sciences, IEEE Computer Society, Washington, DC, USA.

[8] Espinosa, J., Slaughter, S., Kraut, R., and Herbsleb, J. 2007. Team Knowledge and Coordination in Geographically Distributed Software Development. J. Manage. Inf. Syst. 24, 1 (Jul. 2007), 135-169. DOI= http://dx.doi.org/10.2753/MIS0742-1222240104.

[9] Galho, T. S.; Moraes, S. M. W. (2004) Categorização Automática de Documentos de Texto utilizando Lógica Difusa. Logos (Rio de Janeiro), Canoas, v. 15, n. 1, p. 91-104, 2004.

[10] Herbsleb, J. and Mockus, A. (2003) “An empirical study of speed and communication in globally distributed software development”, IEEE Transactions on Software Engineering, volume 29, number 6, pages 481-494.

[11] Herbsleb, J. D. (2007). Global software engineering: The future of socio-technical coordination. FOSE '07: 2007 Future of Software Engineering, pages 188–198.

[12] Hogan, B. (2006) “Lessons Learned from an eXtremely Distributed Project”, In Proceedings of the conference on AGILE 2006, publisher IEEE Computer Society

[13] Jang, C.; Steinfield, C. and Pfaff, B. (2002). “Virtual team awareness and groupware support: an evaluation of the teamSCOPE system”. International Journal of Human Computer Studies, Academic Press, Inc., 56, 109-126.

[14] Kagdi, H. H.; Hammad, M. and Maletic, J. I. "Who can help me with this source code change?" ICSM, IEEE, 2008, 157-166.

[15] Kitchenham, B., 2004. In: Procedures for Undertaking Systematic Reviews. Joint Technical Report, Computer Science Department, Keele University (TR/SE-0401) and National ICT Australia Ltd (0400011T.1).

[16] Layman, L. et al (2006) “Essential communication practices for Extreme Programming in a global software development team”, In journal Information and Software Technology, volume 48, number 9, pages = 781-794

[17] McDonald, D. W. and Ackerman, M. S. (2000) "Expertise recommender: a flexible recommendation system and architecture", In CSCW ’00: Proceedings of the 2000 ACM conference on Computer supported cooperative work, Seiten 231–240, New York, NY, USA. ACM.

[18] Nguyen, T., Wolf, T., and Damian, D. 2008. Global Software Development and Delay: Does Distance Still Matter?. In Proceedings of the 2008 IEEE international Conference on Global Software Engineering (August 17 - 20, 2008). ICGSE. IEEE Computer Society, Washington, DC, 45-54. DOI= http://dx.doi.org/10.1109/ICGSE.2008.39

[19] Parvathanathan, K. et al (2007), “Global Development and Delivery in Practice Experiences of the IBM Rational India Lab”. IBM RedBooks, International Technical Support Organization – IBM

[20] Perry, D. E., Staudenmayer, N., and Votta, L. G. 1994. People, Organizations, and Process Improvement. IEEE Softw. 11, 4 (Jul. 1994), 36-45.

[21] Sim, Y. and Crowder, R. (2004). "Evaluation of an Approach to Expertise Finding", In PAKM, Seiten 141–152

[22] Trindade, C. C., Moraes, A. K. O., and Meira, S. R. L. (2008). Comunicação em equipes distribuídas de desenvolvimento de software: Revisão sistemática. In ESELAW '08: Proceedings of the 5th Experimental Software Engineering Latin American Workshop.

[23] Wives, L. K. (1999) Um Estudo sobre Agrupamento de Documentos Textuais em Processamento de Informações não Estruturadas Usando Técnicas de Clustering. Dissertação de Mestrado, PPGC/UFRGS, Porto Alegre (RS).

[24] Ye, Y.; Nakakoji, K. and Yamamoto, Y. (2007). "Reducing the cost of communication and coordination in distributed software development", Lecture Notes in Computer Science, 2007, LNCS 4716, 152 – 169.