

Presley: uma Ferramenta de Recomendação de Especialistas no Contexto de Código-Fonte para Apoio à Colaboração em Desenvolvimento Distribuído de Software

Cleyton C. Trindade¹, Alan K. O. Moraes^{1,2}, Yuri A. M. Barbosa¹,
Jones O. Albuquerque³, Silvio R. L. Meira^{1,4}

¹Centro de Informática – UFPE, ²Departamento de Informática – UFPB,

³Departamento de Estatística de Informática – UFRPE e

⁴C.E.S.A.R – Centro de Estudos e Sistemas Avançados do Recife

cct@cin.ufpe.br, alan@di.ufpb.br, yamb@cin.ufpe.br,
joa@deinfo.ufrpe.br, silvio@cesar.org.br

***Resumo.** Várias empresas têm adotado o desenvolvimento distribuído de software na busca de novos mercados, de menores custos de desenvolvimento de software e de melhor qualidade dos seus produtos, utilizando a mão-de-obra disponível globalmente. Porém a dispersão das equipes – principalmente a assincronia temporal – dificulta a colaboração entre seus membros, provocando demora na resolução de problemas, pois nem sempre é claro quem é o especialista em determinada parte do software, principalmente, quando esta pessoa está localizada remotamente. Este artigo apresenta uma ferramenta que auxilia o trabalho colaborativo de times distribuídos através da identificação e recomendação dos especialistas em determinadas porções do código-fonte, objetivando diminuir os atrasos causados pela comunicação assíncrona na fase de construção.*

1. Introdução

Muitas das dificuldades ainda enfrentadas por equipes co-localizadas (onde os participantes estão todos no mesmo local) também são encontradas no Desenvolvimento Distribuído de Software (DDS), por exemplo, grande esforço para realizar mudanças [de Souza et al., 2004]. Falhas na comunicação entre os usuários e os desenvolvedores, e entre os próprios desenvolvedores é uma das causas [Ågerfalk and Fitzgerald, 2006]. Em projetos no contexto DDS, a comunicação é realizada em menor quantidade e com menor eficiência [Herbsleb, 2007]. Por exemplo, quando há grande separação temporal, os equívocos gerados por requisitos não claros ou a dificuldade em realizar mudanças têm forte impacto nos cumprimentos de prazos, levando-se, em geral, 2,5 vezes mais tempo para serem identificados e corrigidos em DDS quando comparado com o tempo necessário para projetos co-locados [Espinosa and Pickering, 2006].

Para melhor compreender o problema, realizou-se uma Revisão Sistemática da Literatura [Kitchenham, 2004] a fim de 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 utilizados para minimizar este problema [Trindade et al., 2008]. Entre os pontos destacados pela revisão, a falta de percepção mostrou-se um fator de suma preocupação no DDS, pois

está presente em diversos aspectos na comunicação entre times. Define-se aqui percepção como a capacidade de alguém situar-se durante a execução do projeto e conseguir localizar facilmente as pessoas relevantes para ajudá-lo durante alguma atividade.

Um dos aspectos mais prejudicados com a fraca percepção entre os participantes das equipes é a identificação de especialistas em determinadas porções do código-fonte, que sejam capazes de ajudar os demais membros do time quando surgem dúvidas na fase de construção do software, conseqüentemente, tornando lento o início do processo de colaboração entre os integrantes da equipe [Herbsleb, 2007]. As equipes podem ter um tempo de sobreposição de horário de trabalho muito curto, então a identificação da pessoa mais provável a responder mensagens de dúvidas aponta ser uma grande oportunidade para reduzir os atrasos gerados na comunicação assíncrona entre equipes distribuídas.

Presley, a ferramenta proposta, provê meios para diminuir as deficiências na identificação de especialistas, nos termos definidos no parágrafo anterior, através de mecanismos de inferência e recomendação das pessoas mais qualificadas para atender o membro da equipe que esteja com dificuldades durante a implementação do projeto.

O restante deste trabalho está organizado da seguinte forma: a Seção 2 destaca as características da ferramenta; a Seção 3 introduz o projeto arquitetural adotado; a Seção 4 descreve o funcionamento do mecanismo de inferência; a Seção 5 apresenta os trabalhos relacionados e, por fim, são apresentadas as considerações finais na seção 6.

2. Características do Presley

Presley utiliza o framework teórico descrito por Ye e colegas [2007]. O framework constrói redes de relacionamentos entre os elementos envolvidos na realização do projeto, utilizando-as como entradas de uma heurística para selecionar os especialistas aptos a responderem os problemas enfrentados pelos desenvolvedores na fase de implementação. Os nós da rede, representando código-fonte, documento e desenvolvedor, mostram o relacionamento entre as pessoas e as informações geradas durante o projeto, como apresentado na Figura 1.

Desta forma, o Presley igualmente recebe como parâmetros o código-fonte, os artefatos e registros de comunicação entre os pares no projeto. A ferramenta fornece um canal próprio e específico de comunicação que também será considerado na análise de identificação de especialistas, conforme será detalhado mais adiante.



Figura 1: Rede de atores de um projeto de software, extraído de [Ye et al., 2007]

3. Arquitetura

Presley é um software de arquitetura cliente-servidor. A Figura 2 ilustra a arquitetura da ferramenta e seus principais componentes.

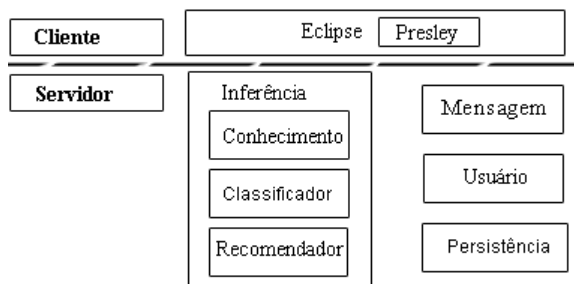


Figura 2: Framework Arquitetural da Ferramenta

O cliente é um plug-in para a IDE Eclipse responsável por coletar e enviar as informações fornecidas pelos usuários para o servidor, através de Java RMI, sem retirar a atenção visual do usuário sobre o código-fonte a ser implementado; tornando-se mais integrada e transparente aos desenvolvedores, como apresentado na Figura 3.

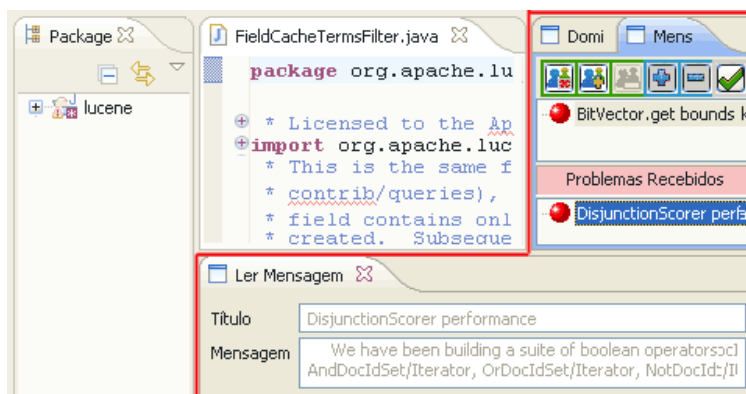


Figura 3: Cliente Presley Integrado a IDE Eclipse

A parte pertencente ao servidor é composta dos seguintes componentes: **Usuário**, **Mensagem**, **Inferência** e **Persistência**. O papel de cada componente é explicado a seguir.

O componente **Usuário** é o responsável por administrar as pessoas envolvidas no projeto e usuárias da ferramenta. Dentre as informações de identificação do usuário, este serviço solicita seu grau de conhecimento prévio sobre os tópicos do domínio do projeto.

O componente **Mensagem** permite a troca de mensagens entre os desenvolvedores. São anexados o conteúdo dos comentários das classes Java referenciadas e relacionados a cada mensagem. Este componente também permite selecionar as contribuições consideradas como úteis na solução de problemas e encaminhar esta informação ao componente Inferência, melhorando os resultados da ferramenta.

O componente **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 documento do projeto a um deles. O componente **Classificador** identifica as palavras-chaves de um documento ou mensagem. 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. Os detalhes da inferência serão descrito na próxima seção.

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.

4. Mecanismo de Inferência

O processo de identificação dos especialistas inicia-se quando um desenvolvedor (remetente) envia uma mensagem de dúvida no Presley. A ferramenta encarrega-se de detectar as classes citadas na mensagem e busca quais outras classes fazem referência direta a elas, compondo o conjunto de classes envolvidas na requisição de ajuda. Através do componente Conhecimento, a ferramenta identifica o conhecimento contido na mensagem através da análise do grau de similaridade entre seu conteúdo (título, conteúdo e comentários dos códigos-fonte) e as palavras-chave (obtidas pelo componente Classificador) que compõem cada conhecimento no banco de dados. Para classificar as mensagens recebidas, uma técnica de similaridade difusa [Galho and Moraes, 2004] foi adaptada para determinar o grau de semelhança da lista de palavras-chave da mensagem de dúvida com a lista de palavras-chave de cada conhecimento. Para cada palavra-chave comum entre as listas, calcula-se o grau de igualdade de seus escores de relevância através da função apresentada na Figura 3. As variáveis a e b representam os escores de relevância de cada.

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

Onde:

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

$$\bar{a}, \bar{b} = 1 - a \text{ ou } 1 - b;$$

$$\wedge = \min.$$

Figura 3: Fórmula do cálculo do grau de igualdade [Galho and Moraes, 2004]

Ao término dos cálculos de igualdade entre as palavras-chave, calcula-se o grau de similaridade entre a mensagem e um dado conhecimento pela fórmula da Figura 4. Este cálculo é repetido para 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 inicia a busca dos usuários que mais enviaram mensagens referentes ao conhecimento identificado no passo anterior e que tenham melhor relacionamento com o usuário remetente (rede social), isto é, aqueles que já tenham registro de correspondência anterior entre eles. O desenvolvedor recebe um ponto para cada mensagem enviada sobre o conhecimento. Em seguida, obtém-se o nível de participação dos desenvolvedores nas classes envolvidas através do histórico da ferramenta de controle de versões. Para cada alteração em uma destas classes, registra-se mais um

ponto. O grau de conhecimento fornecido no cadastro do usuário é então adicionado na pontuação. Os desenvolvedores que obtiverem a maior pontuação no conhecimento serão recomendados ao remetente e receberão o problema a ser solucionado. A ordenação será baseada na rede social do remetente.

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

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.

Figura 4: Fórmula da média por operadores “fuzzy” [Galho and Moraes, 2004]

Em uma simulação preliminar com o projeto Apache Lucene-Java, compreendendo o período de 2004 a 2009, obteve-se uma porcentagem de 80% de acerto nas suas recomendações. Este resultado foi alcançado a partir da seleção aleatória de 20 mensagens da lista de discussão do projeto, onde 16 casos mostraram que ao menos um desenvolvedor reconhecido pelo sistema como especialista realmente respondeu a solicitação de ajuda na lista do projeto.

5. Trabalhos Relacionados

O STeP_IN [Ye et al., 2007] tem como objetivo recomendar um conjunto de especialistas em um determinado assunto do projeto no nível do código-fonte. Essa lista é formada a partir do código-fonte e da comunicação realizada durante o desenvolvimento do projeto, porém não consideram seu conteúdo. 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.

Outro trabalho relacionado é o xFinder [Kagdi et al., 2008] que também tem como objetivo encontrar os especialistas no código-fonte implementado no projeto. Para isso, a recomendação dos especialistas toma como base a relação entre cada código-fonte e os desenvolvedores, extraíndo o número de alterações, a quantidade de dias trabalhados e quando foi o último dia de trabalho do desenvolvedor no arquivo. Por meio dessas informações, os especialistas mais adequados para cada arquivo são selecionados em função do que lhe é mais útil no momento.

Ambas as ferramentas demonstram bons resultados na recomendação de especialistas: 75% para Ye e colegas; entre 42% e 82%, 62,65% na média, para Kagdi e colegas. Presley consegue excelentes resultados (80%) ponderando os esforços prévios de comunicação no time de desenvolvimento, isto é, considerando o conteúdo das mensagens além da rede de contatos estabelecida por Ye e colegas (2007).

6. Considerações Finais

O Desenvolvimento Distribuído de Software vem ganhando destaque em projetos de grande porte, porém a dispersão das equipes – principalmente a assincronia temporal – dificulta a colaboração entre seus membros, ocasionando demora na resolução de

problemas, pois nem sempre é claro quem é o especialista naquela porção do software, principalmente, quando esta pessoa está localizada remotamente. A ferramenta Presley agiliza o processo de descoberta dos especialistas a partir das descrições das dificuldades enfrentadas, do próprio código-fonte e do histórico de comunicação do projeto. Existem parâmetros ainda não adotados que têm o potencial de melhorar a precisão das recomendações, por exemplo, a modelagem temporal consideraria as idades da comunicação e do histórico de versões, dando pesos diferenciados (menores) para eventos mais antigos, assim como feito por Kagdi e colegas (2008).

A ferramenta está disponível para download no endereço <http://collaborare.di.ufpb.br/presley-20091.zip> sob a licença GNU LGPL.

7. Agradecimentos

Aos alunos de Engenharia de Software, turma 2008.1, de Ciência da Computação da UFPB, que auxiliaram no desenvolvimento desta ferramenta.

Referências

- Ågerfalk, P. J. and Fitzgerald, B. (2006). Flexible and distributed software processes: old petunias in new bowls? *Commun. ACM*, 49(10).
- Espinosa, A. J. and Pickering, C. (2006). The effect of time separation on coordination processes and outcomes: A case study. In *HICSS '06: Proceedings of the 39th Annual Hawaii International Conference on System Sciences*, Washington, DC, USA. IEEE Computer Society.
- 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.
- Herbsleb, J. D. (2007). Global software engineering: The future of socio-technical coordination. In *FOSE '07: 2007 Future of Software Engineering*, pages 188–198, Washington, DC, USA. IEEE Computer Society.
- 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)
- 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
- Kagdi, H.; Hammad, M.; Maletic, J.I. (2008) Who can help me with this source code change? In the 24th IEEE International Conference on Software Maintenance (ICSM'08), Beijing China, Sept. 28 - Oct. 4, 2008, pp. 157-166
- de Souza, C. R., Redmiles, D., Cheng, L., Millen, D., and Patterson, J. 2004. Sometimes you need to see through walls: a field study of application programming interfaces. *CSCW '04: Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*. ACM, New York, NY, 63-71.
- 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*.