Usando Recursos de Visualização Enriquecidos com Elementos de Percepção para a Compreensão de Software em um Ambiente de Desenvolvimento Distribuído

Carlos F. R. Conceição[1], Glauco de Figueiredo Carneiro[1], José Maria N. David[2]

[1] Departamento de Sistemas e Computação, Universidade Salvador (UNIFACS) - BA

[2] Departamento de Ciência da Computação, Universidade Federal de Juiz de Fora - MG

cfabioramos@unifacs.br, glauco.carneiro@unifacs.br, jose.david@ufjf.edu.br

Abstract. *This paper describes the use of visualization resources enriched by perception elements to enhance software comprehension. For this purpose, the visual environment SourceMiner was extended to provide information that describes actions performed by programmers. The goal is to enable the use of perception elements in a distributed environment. A case study was conducted to analyze the effectiveness of this support to software comprehension activities in a distributed environment. The results provide initial evidences that perception elements enhance software comprehension activities performed by geographically distributed teams.*

Resumo. *Este artigo descreve o uso de recursos de visualização enriquecidos com elementos de percepção para potencializar a compreensão de software. Para esta finalidade, o ambiente visual SourceMiner foi estendido para prover informações que descrevem ações executadas pelos programadores. O objetivo é permitir o uso de elementos de percepção em ambientes distribuídos. Um estudo de caso foi conduzido para analisar a efetividade deste suporte a atividades de compreensão de software em ambientes distribuídos. Os resultados apresentam evidências iniciais de que os elementos de percepção potencializam as atividades de compreensão de software executadas por equipes distribuídas geograficamente.*

1. Introdução

A execução de atividades de compreensão de software em ambientes de desenvolvimento distribuído necessita de uma comunicação eficiente. Quando isto não ocorre, tem-se que a interação entre os programadores fica prejudicada. Sendo a comunicação o principal elemento para promover a percepção, quando barreiras são criadas para ela, então a percepção também é prejudicada. A percepção é um elemento relevante neste cenário pelo fato de contextualizar os programadores para a execução de suas atividades (Dourish e Bellotti, 1992). Ela permite identificar quem está trabalhando no projeto, o que eles estão fazendo, em quais artefatos estão ou estavam manipulando, e como seus trabalhos poderão impactar outros trabalhos (Storey et al., 2006). Neste contexto, tem-se que a ausência de suporte a elementos de percepção no desenvolvimento distribuído de software pode prejudicar a execução de atividades de compreensão (Conceição, 2012).

O Collaborative SourceMiner (Conceição, 2012) é o resultado da combinação de um ambiente interativo baseado em múltiplas visões com elementos de percepção para apoiar a compreensão de software no desenvolvimento distribuído. O objetivo é

fornecer informações que permitam aos integrantes das equipes conhecerem, através do ambiente, o que os demais pesquisaram, manipularam ou alteraram em um determinado projeto de software.

Pesquisas prévias desenvolveram soluções para apoiar a percepção em uma infraestrutura distribuída de desenvolvimento e manutenção de sistemas. Por exemplo, Sarma et al. (2003) desenvolveram o conceito de ponderação. Ponderação é o que deve ser levado em consideração para promover a percepção nas interações dos programadores. Cada interação tem seu nível de impacto no estado do projeto compartilhado, e o que é ponderado é compartilhado com o objetivo de promover a percepção. O resultado desse trabalho é uma ferramenta chamada Palantír. O ProjectWatcher é o resultado de uma pesquisa elaborada por Schneider et al. (2004), cujo objetivo é disponibilizar históricos de interações. Além disso, provê diferentes visualizações de quem está ativo no projeto, em quais artefatos e atividades eles estão atuando. O Lighthouse (Silva et al., 2006) cria visualizações com informações coletadas dos espaços de trabalho dos programadores. Neste trabalho foi aplicado o conceito de *Emerging Design*, que é uma representação atualizada do projeto tal como existe no código dos programadores.

Apesar das soluções propostas pelas pesquisas em desenvolvimento distribuído de software, pouco tem sido explorado no que diz respeito à utilização de ambientes interativos baseados em múltiplas visões. Para tanto, a solução proposta pelo Collaborative SourceMiner considera que programadores geograficamente dispersos necessitam de um conjunto diversificado de visões para apoiar a atividade de compreensão.

Este artigo está organizado da seguinte forma: a seção 2 apresenta os elementos relevantes para a contextualização deste artigo: desenvolvimento distribuído de software (DDS), percepção e compreensão de software. A seção 3 apresenta o modelo conceitual proposto. A seção 4 apresenta um estudo de caso para analisar como programadores identificam anomalias de modularidade de software em um cenário distribuído. Por fim, a seção 5 apresenta as considerações finais.

2. Compreensão e Percepção em Desenvolvimento Distribuído de Software

O desenvolvimento distribuído de software tem como um dos principais objetivos o ganho de produtividade, a redução de custos e a melhoria na qualidade do software (Prikladnicki e Audy, 2007). Entretanto, dentre as dificuldades inerentes ao desenvolvimento de software, tem-se que a forma como se dá a comunicação entre os participantes como uma questão importante a ser analisada. Sendo a percepção dependente da comunicação, tem-se que a percepção pode ser dificultada em ambientes de desenvolvimento distribuído.

A compreensão de software consiste na obtenção de conhecimento das funcionalidades, estrutura e comportamento de um sistema de software (Mayrhauser e Vans, 1993). No desenvolvimento distribuído, as atividades de compreensão podem ser dificultadas, por exemplo, pelos seguintes motivos: determinar quem possui conhecimento sobre diferentes partes do projeto (Zimmermann e Selvin, 1997), e comunicar com outros integrantes da equipe que atuam em horários diferentes (Herbsleb et al., 1999).

Elementos de visualização de software têm sido propostos na literatura para apoiar

a compreensão de software (Pacione, 2004; Lintern et al., 2003). Entretanto, o que se tem verificado é a necessidade de suporte nestas propostas à colaboração.

Conforme evidenciado por Gutwin e Greenberg (2002), o espaço de trabalho tem um papel importante durante a colaboração, mantendo o conhecimento sobre as interações dos outros, resultando em informações para a percepção. Omoronyia (2009) propõe que os benefícios da divisão das atividades de desenvolvimento distribuído podem ser encontrados capturando as trilhas de interações que ocorrem dentro dos espaços de trabalho. Essas trilhas podem ser construídas quando programadores atuam sobre suas tarefas de desenvolvimento diário deixando para trás vestígios de histórico daquilo que realizaram.

Em ambientes de programação distribuída, a utilização de sistemas colaborativos pode ser útil para apoiar a coleta e disponibilização de informações a respeito das interações no espaço de trabalho. Tais informações podem ser integradas com o conhecimento existente para manter um senso de percepção. Este senso permite, por exemplo, identificar em qual artefato o outro integrante está atuando e o que ele está fazendo. Quando este conhecimento se associa àqueles adquiridos com a interpretação de metáforas visuais, têm-se como resultado a geração de significados compartilhados sobre os artefatos para potencializar atividades de compreensão de software (Conceição, 2012).

3. O Modelo Conceitual Proposto

A Figura 1 ilustra o modelo conceitual para a solução proposta neste trabalho. A **parte A** da figura indica a atuação dos programadores na compreensão do código fonte utilizando o ambiente de desenvolvimento de software (ADS) Eclipse. O ADS é utilizado em conjunto com o plug-in Collaborative SourceMiner, representado na figura pelo círculo vermelho. A **parte B** ilustra o uso combinado das visões enriquecidas com dados obtidos dos elementos de percepção. Estes dados são provenientes das "mensagens de usuário" e "mensagens de sistema". A **parte C** indica que os dados da aplicação analisada obtidos do modelo Java (*Java Model*) poderão ser enriquecidos por métricas relacionadas ao tamanho, complexidade ciclomática, acoplamento, entre outras. A **parte D** ilustra que a interação com os cenários visuais é viabilizada através dos filtros de controle. O resultado esperado é um conjunto de visões enriquecidas com elementos de percepção para apoiar as atividades de compreensão de software, conforme ilustrado na **parte E**.

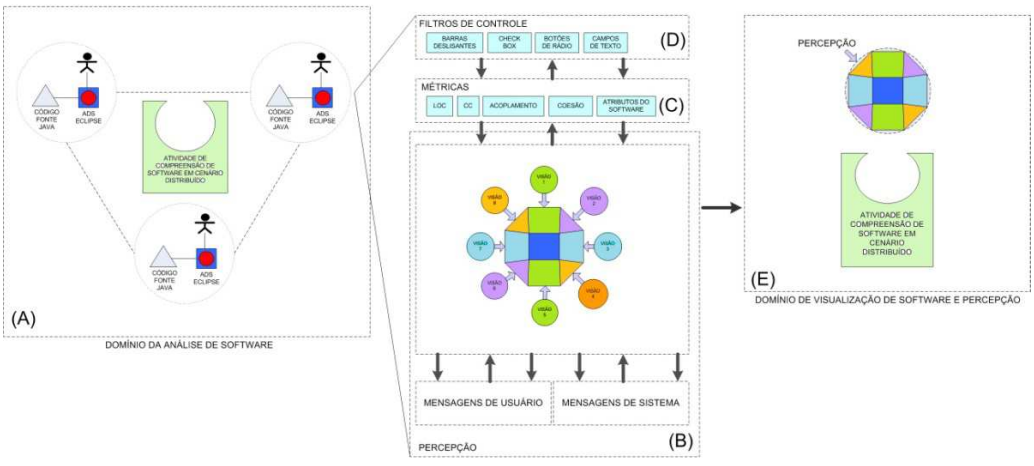

Figura 1. O Modelo Conceitual para o Collaborative SourceMiner

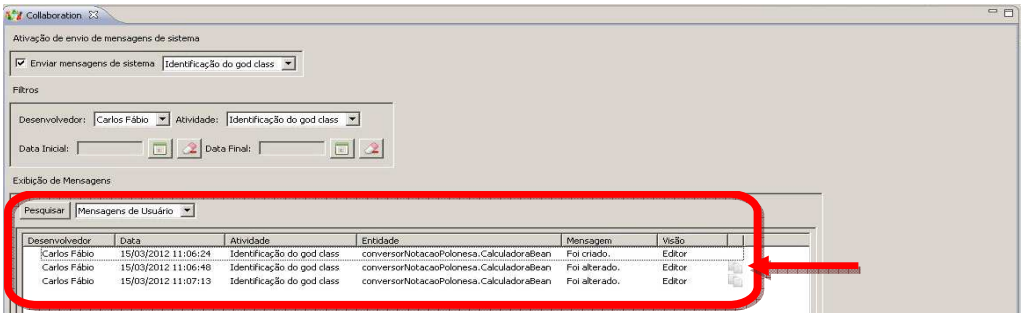

Figura 2. Visão Collaboration Detalhando as Ações de um Programador

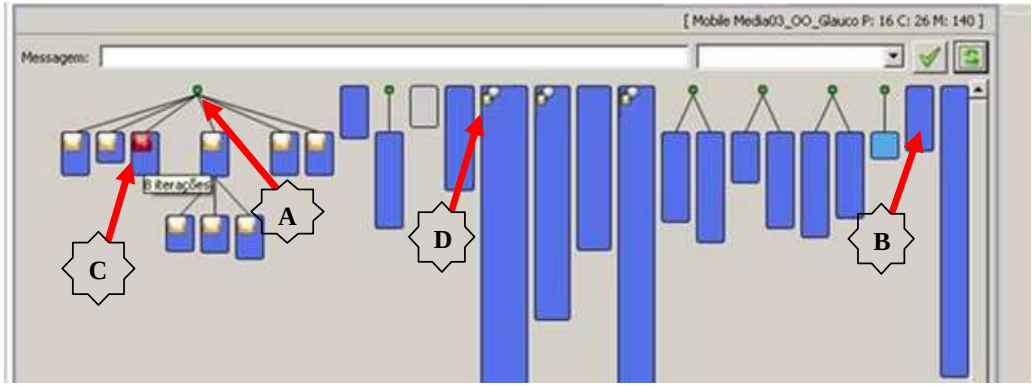

Figura 3. Visão Polimétrica Exibindo Mensagens de Usuário e Sistema

O modelo possibilita a comunicação semi-síncrona. As interações ocorrem tanto através de mensagens criadas por programadores como também de mensagens geradas pelo sistema. O objetivo das mensagens de usuário é registrar informações julgadas relevantes para contextualizar os programadores. As mensagens de sistema são sequências de eventos coletadas a partir de ações executadas pelos programadores. A partir do momento que um programador indica o início da execução de uma atividade, suas ações serão coletadas e enviadas para um servidor.

No quadro em destaque na Figura 2 são apresentados exemplos de mensagens de sistema. Cada registro exibe as seguintes informações: programador, data e horário, atividade, entidade (classe, interface, ou método) e visão utilizada. Estas informações servem para caracterizar as ações executadas por um determinado programador. Quando a ação executada representa uma modificação na composição de uma entidade, o ícone apontado pela seta é exibido. Este ícone pode ser selecionado para que sejam exibidas as versões anterior e posterior à alteração. Estas informações têm o objetivo de contextualizar os programadores que atuam em uma mesma atividade.

A Figura 3 exibe a visão polimétrica (Carneiro, 2011) que representa a hierarquia de herança das entidades (classes e interfaces) de um sistema. As interfaces são representadas pelos círculos esverdeados (seta A da figura). As classes são representadas por retângulos azuis (seta B da figura). Este é um exemplo de como as visões podem ser enriquecidas com elementos de percepção. Para esta finalidade, os ícones indicados pelas setas C e D são os recursos visuais utilizados para indicar a existência de mensagens associadas à entidade. Neste caso, a seta C indica o ícone específico para representar mensagens de sistema, enquanto que a seta D indica o ícone

que representa mensagens de usuário. De acordo com a figura, os ícones que representam as mensagens de sistema podem variar na tonalidade da cor vermelha. Ícones mais escuros destacam as entidades que um determinado programador mais interagiu. Desta forma, quem visualiza este cenário tem uma indicação inicial das entidades que a princípio estão mais relacionadas a uma atividade.

4. Um Estudo de Caso com o Collaborative SourceMiner

Um estudo de caso foi realizado com o objetivo de analisar a seguinte pergunta de pesquisa: "Como os elementos de apoio à percepção presentes no Collaborative SourceMiner auxiliam a compreensão de software considerando que os programadores atuam de forma colaborativa em locais geograficamente distribuídos?". O estudo contou com seis participantes, divididos em duas equipes com três em cada, não havendo interação entre as equipes. Cada equipe foi solicitada a identificar as anomalias de modularidade de software *Feature Envy*[1] (Fowler, 1999), *God Class*[2] (Riel, 1996) e *Divergent Change*[3] (Fowler, 1999). A aplicação selecionada foi o Mobile Media (MobileMedia, 2006). Todos os participantes selecionados para o estudo tinham conhecimento da linguagem Java e já tinham utilizado o ADS Eclipse. Durante a execução das atividades solicitadas, eles responderam um questionário para a coleta de dados. Estes dados foram utilizados nas avaliações quantitativa e qualitativa.

4.1. Análise e Discussão dos Resultados

A Tabela 1 apresenta os valores de precisão (p) e cobertura (c) de cada participante das equipes na identificação das anomalias de modularidade de software. A precisão quantifica a taxa de anomalias de modularidade corretamente identificadas pelo número de anomalias de modularidade detectadas. A cobertura quantifica a taxa de anomalias de modularidade corretamente identificadas pelo número de anomalias de modularidade da lista de referência já adotada em outros estudos (Carneiro, 2011). PA1, PA2 e PA3 representam os participantes 1, 2 e 3 da primeira equipe. PA4 e PA6 representam os participantes 1 e 3 da segunda equipe. PA5 não respondeu o questionário da forma solicitada, por este motivo seus valores não são apresentados neste estudo.

Tabela 1: Resultados por Participante das Equipes

	PA1		PA2		PA3		PA4		PA6	
	c	p	c	p	c	p	c	p	c	p
GC	0,9	1	0,8	0,9	0,9	0,7	0,2	0,7	0,2	0,7
DC	0,2	0,6	0,1	0,4	0,1	0,1	0,1	0,1	0,1	0,4
FE	0,4	0,3	0,1	0,1	0,2	0,2	0,2	0,4	0,2	0,6

A análise dos dados apresentados na Tabela 1 permite concluir que a maior variação

[1] *Feature Envy* ocorre quando um trecho de código de uma classe parece fazer mais parte de outra classe do que aquela na qual está contido (Fowler, 1999).
[2] *God Class* é caracterizado pela não coesão de comportamento e forte tendência de uma classe em atrair mais funcionalidades (Riel, 1996).
[3] *Divergent Change* ocorre quando uma classe necessita ser alterada frequentemente por diferentes razões (Fowler, 1999).

de precisão e cobertura ocorreu na equipe 1. Neste caso, o valor da precisão obtido por PA1 para *Divergente Change* foi de 0,6 e de 0,1 para PA3. Já na segunda equipe ocorreram duas variações nos valores de precisão e cobertura. A primeira relacionada à anomalia *Divergent Change* de 0,1 (PA4) para 0,4 (PA6). A segunda relacionada à anomalia *Feature Envy* de 0,4 (PA4) para 0,6 (PA6). Esta variação pouco acentuada pode ser atribuída à colaboração que ocorreu entre os participantes. Outra observação relevante foi a de que a anomalia *God Class* com os maiores valores de precisão e cobertura foi a que apresentou reduzida variação se considerados os valores de cada participante de cada equipe. A Tabela 2 apresenta as indicações dos participantes da primeira equipe para a identificação de *God Class*. Estão em destaque, sublinhadas e em negrito, as indicações convergentes dos participantes. Através dos relatos dos participantes, das evidências da comunicação que ocorreu através do ambiente, e da sequência de ações registradas foi possível identificar que os elementos de percepção disponíveis na ferramenta foram fundamentais para o alto nível de convergência.

Tabela 2: Indicações dos Participantes da Eq.1 de Ocorrências de God Class

Participante 1				
Versão 3	**Versão 4**	**Versão 5**	**Versão 6**	**Versão 7**
BaseController	**BaseController**	**ImageAcessor**	PhotoController	**MediaController**
ImageAcessor	**ImageAcessor**		**ImageAcessor**	
Participante 2				
BaseController	**BaseController**	**ImageAccessor**	**ImageAccessor**	**MediaController**
ImageAccessor	**ImageUtil**	AlbumData		
Participante 3				
BaseController	**BaseController**	**ImageAcessor**	**ImageAcessor**	**MediaAcessor**
InvalidImageDataException	InvalidImageDataException	AlbumData		**MediaController**
ImageAcessor	**ImageAcessor**			
PhotoController				

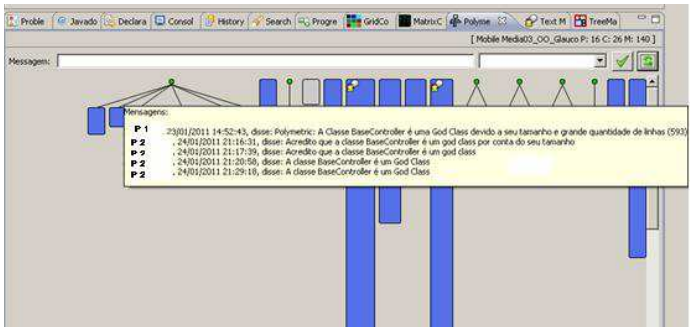

Figura 4. Comunicação entre os Participantes Utilizando a Visão Polimétrica

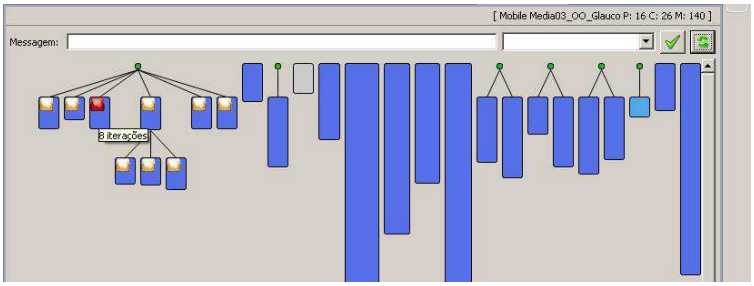

Figura 5. Visão Polimétrica Exibindo Mensagens de Sistema

Os trechos de conversa entre os participantes apresentados na Figura 4 revelam evidências da comunicação durante a execução das atividades. A Figura 5 exibe a visão polimétrica vista por PA2 na identificação de *Feature Envy*. Para obter esta visão PA2

executou uma consulta utilizando as seguintes opções de filtro: participante-PA1 e atividade-*Feature Envy*. Tendo como base esta visão, PA2 relatou: “após perceber que PA1 interagiu mais vezes com a classe *UnavailablePhotoAlbumException* acessei-a para verificar se de fato se tratava de uma ocorrência de *Feature Envy*.” O resultado da análise foi que a classe em questão não era uma ocorrência de *Feature Envy*.

Após a conclusão das atividades, PA1 relatou: “Os comentários dos outros participantes me auxiliaram a perceber algumas características de classes antes não observadas por mim.”. PA2 relatou: “Quando os outros participantes já haviam identificado as classes, eu procurava outras classes, ou conferia as classes identificadas. Era como resolver um exercício em grupo. Fiquei com algumas dúvidas, mas aprendi através do acesso às ações e respostas registradas por meus colegas”. Já PA3 relatou: “As mensagens incisivas, principalmente as de PA1, foram fundamentais para tirar possíveis dúvidas que eu tive na execução da atividade”. PA4 também relatou: “A indicação da classe *BaseController* foi em concordância com a resposta de PA6.”. Estes comentários indicam evidências iniciais de que o enriquecimento das representações visuais com as dicas dos participantes contribuiu para a geração de um entendimento compartilhado.

As seguintes observações foram obtidas considerando a pergunta de pesquisa deste estudo e a análise dos resultados. **Observação 1**: durante a execução das atividades, os participantes se comunicaram e, na maioria das vezes, convergiram em suas indicações. **Observação 2**: os resultados apresentam evidências iniciais de que os participantes inicialmente buscavam no ambiente a existência de indícios que os fizessem conhecer a respeito das indicações dos outros. Quando encontravam, faziam uma análise dessas indicações para, então, elaborar as suas. **Observação 3**: os resultados apresentam evidências iniciais de que, devido aos elementos de percepção presentes na ferramenta, os participantes adotaram estratégias parecidas na identificação das anomalias. Ou todos adotaram uma estratégia otimista, ou então adotaram uma estratégia pessimista.

Ameaças à Validade do Estudo. O tamanho e a complexidade do objeto de estudo (o Mobile Media) estão aquém daquelas relacionadas às aplicações comerciais típicas. Entretanto, esta aplicação já tem sido utilizada em outros estudos a exemplo de Carneiro (2011) para fins da caracterização do uso de ambientes de visualização de software. O número reduzido de participantes pode comprometer a representatividade dos resultados obtidos e a sua generalização. O número de participantes considerou o equilíbrio entre o custo do estudo e a análise qualitativa dos resultados para derivar as observações.

5. Conclusão

Este artigo apresentou um estudo de caso para caracterizar o uso de recursos de visualização enriquecidos com elementos de percepção para a compreensão de software em um ambiente de desenvolvimento distribuído. O estudo foi conduzido com duas equipes de três participantes cada que atuaram em um ambiente de desenvolvimento distribuído. Para a execução das atividades os participantes utilizaram exclusivamente o Collaborative SourceMiner como suporte à percepção. Os resultados do estudo apresentaram evidências iniciais da efetividade do uso de elementos de percepção para a compreensão de software de forma colaborativa. O modelo conceitual e a implementação da solução proposta estão sendo revistos tendo como referência as oportunidades de melhorias identificadas no estudo descrito neste artigo. A próxima etapa será o planejamento de um novo estudo com foco no suporte à coordenação em

ambientes de desenvolvimento distribuído.

Referências

Carneiro, G. F. (2011) “SourceMiner: Um Ambiente Integrado Para Visualização Multi-Perspectiva De Software”. 229 f. Tese de Doutorado em Ciência da Computação. Universidade Federal da Bahia.

Conceição, C.F.R. (2012) “Analisando o Uso de Elementos de Percepção Para Atividades de Compreensão de Software em um Ambiente de Desenvolvimento Distribuído”. Dissertação de Mestrado, UNIFACS, Salvador.

Dourish, P.; Bellotti, V. (1992) “Awareness and coordination in shared workspace”, Conference on Computer-Supported Cooperative Work. pp. 107-114, Toronto, Canada, Nov.

Fowler, M. (1999) “Refactoring: Improving the Design of Existing Code”. Addison-Wesley Professional.

Herbsleb, J.; Grinter, R.; Perry, D. (1999) “The geography of coordination: dealing with distance in R&D work”. Proc. ACM SIGGROUP conference on supporting group work.

Gutwin, C.; Greenberg, S. (2002) “A Descriptive Framework of Workspace Awareness for Real-Time Groupware”. Journal of Computer-Supported Cooperative Work. Issue 3-4. p. 411-446.

Lintern R.; Michaud J.; Storey, M.; Wu, X. (2003) “Plugging-in Visualization: experiences integrating a visualization tool with Eclipse”. Proceedings of the 2003 ACM symposium on Software visualization, pp. 47-56, New York.

Mayrhauser, A. V.; Vans, A. M. (1993) “From Code Understanding Needs to Reverse Engineering Tool Capabilities”. Proceedings of the 6th International Workshop on Computer-Aided Software Engineering, pp. 230-239.

MOBILEMEDIA. MobileMedia. Disponível em http://mobilemedia.sourceforge.net. 2006.

Omoronyia, I. (2009) “Using Developer Activity Data to Enhance Awareness during Collaborative Software Development”. University of Strathclyde. Glasgow, p. 50.

Pacione, M. J. (2004) “Software visualisation for object-oriented program comprehension”. In Doctoral Symposium, Proceedings of the 26th International Conference on Software Engineering (ICSE 2004), pp. 63-65, Edinburgh. IEEE / ACM.

Prikladnicki, R.; Audy, J. L. N. (2007) “Um Modelo de Referência para Desenvolvimento Distribuído de Software”. 16 f. Artigo (3º) - Curso de Informática, Departamento de Informática, Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre.

Riel, A. J. (1996) “Object-oriented design heuristics”. Addison-wesley professional.

Sarma, A.; Noroozi, Z.; Hoek, A. (2003) “Palantír: raising awareness among configuration management workspaces”. In International Conference on Software Engineering (ICSE ‘03). IEEE Computer Society. pp. 444-454.

Silva, I. et al. (2006) “Lighthouse: coordination through emerging design”. In ETX 2006 (OOPSLA Workshop On Eclipse Technology Exchange), pp. 11 – 15.

Storey, M.-A; Cheng, L.-T; Rigby, P.; Bull, R.I. (2006) “Shared Waypoints and Social Tagging to Support Collaboration in Software Development”, In Proceedings of Computer Supported Cooperative Work. Banff, Canada, 195-198.

Zimmermann B.; Selvin, A. M. (1997) “A framework for assessing group memory approaches for software design projects”. Proc. Conference on Designing interactive systems.