ReuseDashboard: Apoiando *Stakeholders* na Monitoração de Programas de Reutilização de Software

Marcelo Palmieri, Marcelo Schots, Cláudia Werner

Programa de Engenharia de Sistemas e Computação (PESC) – COPPE/UFRJ
Caixa Postal 68.511 – 21945-970 – Rio de Janeiro, RJ – Brasil

{palmieri, schots, werner}@cos.ufrj.br

Abstract. *Software reuse programs provide several benefits, such as increase in productivity, decrease of time-to-market, amongst others. However, reuse programs can be hard to be implemented in organizations due to the large amount of information, which is usually not targeted to the stakeholder involved in a reuse task. This leads to cognitive overloading, despite other problems. This work presents ReuseDashboard, a mechanism that aims to support reuse stakeholders by providing relevant information through visual analytics, based on reuse-related metrics. The mechanism supports software reuse programs in both development level (development for and with reuse) and management level.*

Resumo. *Programas de reúso de software proveem diversos benefícios, como o aumento da produtividade, a diminuição do time-to-market, dentre outros. No entanto, a implementação de programas de reúso nas organizações pode ser difícil devido à grande quantidade de informação, que usualmente não é direcionada ao stakeholder envolvido em uma tarefa de reúso. Isto resulta em sobrecarga cognitiva, além de outros problemas. Este trabalho apresenta o ReuseDashboard, um mecanismo que visa apoiar stakeholders de reúso provendo informações relevantes por meio de visual analytics, baseado em métricas relacionadas a reúso. O mecanismo apoia programas de reúso tanto em nível de desenvolvimento (com e para reúso) quanto em nível gerencial.*

1. Introdução e Motivação

A reutilização de software é a prática de desenvolvimento na qual artefatos de software preexistentes ou conhecimento de projetos anteriores são utilizados para a criação de novos produtos de software [Frakes *et al.* 2005]. Tal prática propicia inúmeros benefícios ao longo do processo de desenvolvimento, tais como: (i) a diminuição do esforço de implementação, devido à utilização de um mesmo produto de trabalho por várias vezes, culminando no aumento da produtividade, (ii) a amortização de custos de inspeção e teste, favorecendo o aumento da qualidade, (iii) a redução do tempo de entrega do produto, (iv) a facilitação na definição de padrões (*patterns*) e normas, e (v) a promoção da interoperabilidade e compatibilidade (Frakes *et al.*, 1994).

Entretanto, a reutilização pode ser uma tarefa custosa para o desenvolvimento para ou com reutilização. No desenvolvimento com reutilização, o desenvolvedor precisa conhecer o software reutilizável e suas características, bem como compreender sua execução, de forma a poder reutilizá-lo devidamente. Já no desenvolvimento para

reutilização, é preciso que o software possua algumas características importantes para reutilização, tais como generalidade e flexibilidade, além de estar enquadrado na política de reutilização da organização [Caldiera e Basili, 1991].

Além disso, a implementação de programas de reutilização nas organizações pode ser difícil devido a problemas de gestão, falta de compreensão, falta de incentivos financeiros e sobrecarga cognitiva, dentre outros [Kim e Stohr, 1998]. Muitas vezes isto se deve à grande quantidade de informação a ser analisada e à falta de ferramentas e técnicas de apoio à reutilização de software, que não consideram os diferentes *stakeholders* envolvidos no processo de reutilização e seus interesses em particular.

Diehl (2007) afirma que a compreensão de software é uma atividade complexa, que requer recursos específicos para facilitar o seu desenvolvimento. É neste contexto que técnicas de visualização, combinadas com a utilização de métricas apropriadas, podem facilitar o entendimento do software e seu desenvolvimento, promovendo representações intuitivas e relevantes ao contexto de cada *stakeholder* envolvido no desenvolvimento do software [Lanza e Marinescu, 2006]. Faz-se necessário, no entanto, identificar mecanismos adequados, abstrações apropriadas, e obter evidências sobre o estímulo da percepção humana e habilidades cognitivas [Schots *et al.*, 2012].

O apoio visual na tomada de decisão tem recebido atenção especial da indústria e da academia. Nas últimas décadas, o rápido crescimento de dispositivos de armazenamento de dados junto aos meios de criar e coletar dados influenciaram nossa forma de lidar com a informação [Keim *et al.*, 2008; Schots *et al.*, 2012]. Neste cenário, uma subárea que tem se sobressaído é a de *visual analytics*, considerada "a ciência do raciocínio analítico facilitado por interfaces visuais interativas" [Thomas & Cook, 2006]. Seu objetivo é tornar transparente a forma de o ser humano processar dados e informações, visando o raciocínio analítico [Keim *et al.*, 2008].

A utilização de informações de forma analítica pode ser transportada para o cenário de reutilização de software visando apoiar a atividade de compreensão do software. Por exemplo, a combinação de métricas de software com técnicas de visualização pode fornecer sumarizações dos dados extraídos, exibindo-os de forma agregada através de abstrações visuais intuitivas para cada *stakeholder* do programa de reutilização de software, permitindo que cada *stakeholder* customize as informações a serem apresentadas e suas representações visuais conforme sua necessidade.

Neste sentido, este trabalho apresenta o ReuseDashboard, um mecanismo interativo voltado para programas de reúso, que faz uso de métricas e *visual analytics* visando estimular o envolvimento dos *stakeholders*, provendo informações relevantes a cada papel envolvido em tarefas de reúso. O artigo está organizado da seguinte forma: a Seção 2 exibe trabalhos relacionados, a Seção 3 apresenta o ReuseDashboard, a Seção 4 descreve um cenário ilustrativo, e a Seção 5 contém as considerações finais.

2. Trabalhos Relacionados

Kuipers *et al.* (2007) apresentam um método baseado em ferramentas para a monitoração de software, visando a qualidade do mesmo. Assim como o ReuseDashboard, o trabalho dá ênfase nas tarefas dos *stakeholders*, exibição de visualizações e utilização de métricas. Entretanto, existem algumas limitações, a saber:

(i) a utilização de um modelo fixo para a avaliação de qualidade do software, baseado em poucas métricas, não se adequando às necessidades de cada *stakeholder*, dificultando assim a tomada de decisão; (ii) poucos tipos de visualizações, com informações categorizadas apenas em "muito alto", "alto", "moderado" e "baixo"; e (iii) a impossibilidade de investigar os resultados da análise em um ambiente de desenvolvimento indicando a que trecho do código corresponde um dado, o que facilitaria o desenvolvedor na localização de possíveis problemas.

Plösch *et al.* (2008) apresentam uma abordagem que tem por objetivo apoiar a avaliação de qualidade do software pelos *stakeholders* envolvidos no processo de desenvolvimento. O trabalho foi desenvolvido como um *plugin* do Eclipse, e utiliza um modelo de qualidade baseado no padrão ISO, oferecendo uma grande quantidade de métricas e atributos de qualidade para avaliação, apoiando diferentes *stakeholders*. Para realizar a avaliação de um software, é necessário preencher uma grande quantidade de informações, e vários relatórios são gerados. Além disso, não é mencionado nenhum apoio visual para a representação das informações geradas, dificultando a tomada de decisão. Há pouca flexibilidade na construção de modelos de avaliação (que deve ser feita via código), e todas as informações são apresentadas apenas na IDE, fazendo com que todos os *stakeholders* precisem utilizá-la para obter tais informações, o que pode ser incômodo e não muito familiar para *stakeholders* de nível mais gerencial.

3. ReuseDashboard

O mecanismo proposto neste trabalho, nomeado de ReuseDashboard, visa auxiliar os diversos *stakeholders* do processo de desenvolvimento no acompanhamento de um programa de reutilização de software, provendo informações visuais analíticas. Para isto, a abordagem permite a criação de um plano de avaliação a partir de um conjunto de métricas relacionadas à reutilização, que pode ser utilizado para apoiar atividades de reúso, tais como a identificação de componentes, a avaliação de qualidade do software, a identificação de anomalias etc. As medidas são, então, apresentadas por meio de visualizações configuráveis, a fim de facilitar a análise e tomada de decisão. Além disso, visando fornecer um ambiente heterogêneo que permita que cada *stakeholder* acompanhe o programa de reúso sem sair de suas atividades habituais, o ReuseDashboard é compatível com plataformas *desktop* e dispositivos móveis.

Os passos executados pelo mecanismo são: (i) identificação e extração de métricas relacionadas a reúso a partir de projetos de software, (ii) avaliação das métricas extraídas a partir de um plano de avaliação configurável, e (iii) visualização multidispositivo das métricas extraídas e dos resultados da análise realizada por meio do plano de avaliação. Este mecanismo é uma evolução da abordagem proposta em [Palmieri e Werner 2012], contemplando múltiplos *stakeholders*.

Por se tratar de um mecanismo multidispositivo, o ReuseDashboard está sendo desenvolvido em um ambiente web, visando alcançar tanto plataformas móveis quanto plataformas *desktop*, estando ainda integrado com o ambiente de desenvolvimento Eclipse. O mecanismo também possui a característica de ser independente de sistema operacional. Tais requisitos são importantes para se adequar à realidade de trabalho de cada um dos *stakeholders* envolvidos. Indivíduos com perfil mais técnico (tais como desenvolvedores) podem usufruir da integração do mecanismo com a IDE Eclipse,

enquanto indivíduos com perfil mais gerencial podem utilizar o ReuseDashboard a partir de seus computadores pessoais ou de dispositivos móveis (e.g., *tablets* e *smartphones*) que possuam um navegador (*browser*) disponível.

A arquitetura do ReuseDashboard está dividida em dois módulos principais. O primeiro, chamado MEEP (*Metrics Extraction and Evaluation Plan*), trata-se de um *plugin* da IDE Eclipse, responsável por fazer a extração e a análise das métricas a partir do código fonte de projetos de software. O segundo, denominado Viz (*Visualization*), é um projeto para web, com o objetivo de fornecer, por meio de um *dashboard*, informações analíticas – isto é, informações que já foram pré-analisadas (como por exemplo, média e somatório) – dos dados extraídos e analisados no primeiro módulo.

3.1. Extração de Métricas e Plano de Avaliação

O MEEP fornece uma interface para que os *stakeholders* possam utilizar planos de configuração previamente elaborados (que podem ser customizados) ou criar seus próprios planos. Tais planos são utilizados para analisar projetos em Java contidos na IDE Eclipse. O plano de avaliação se baseia nas estratégias de detecção presentes em [Lanza e Marinescu, 2006] e consiste em um conjunto de associações comparativas entre medidas e limiares (*thresholds*) que funcionam como uma regra que caracteriza uma determinada situação. Caso o resultado de todas as comparações medida-limiar seja verdadeiro, isto indica a ocorrência da situação especificada.

A Figura 1 ilustra um possível plano de avaliação para a identificação de componentes candidatos à reutilização, com base em métricas extraídas de [Cho *et al.*, 2001], com valores customizados ao cenário de uma determinada empresa. Cabe ressaltar que tais valores são fortemente dependentes do tipo e tamanho do projeto. A Figura 2, por sua vez, exibe a tela de seleção e criação de um plano de avaliação.

Figura 1 – Estratégia de detecção (adaptada de [Lanza e Marinescu 2006]) para a identificação de candidatos à reutilização

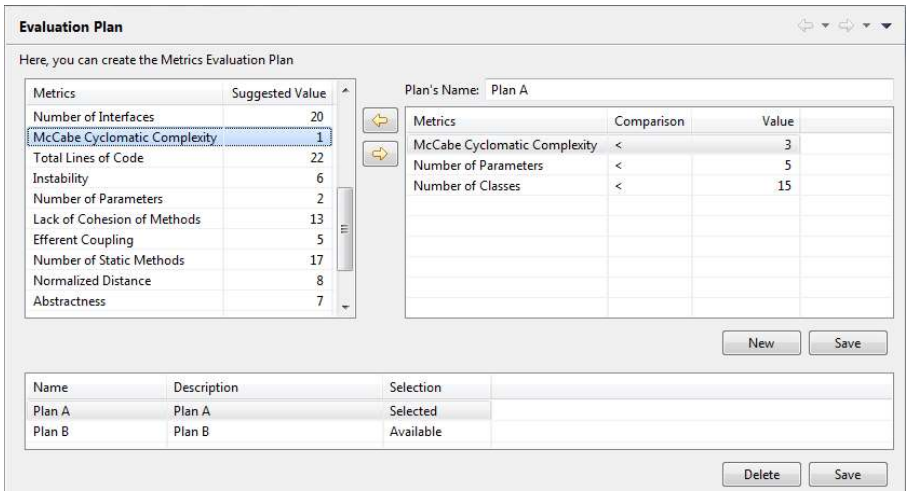

Figura 2 – Exemplo de Plano de Avaliação

Todos os valores extraídos e o resultado da avaliação são repassados para o Viz, que contém um conjunto de gráficos e métricas que são selecionados e combinados de forma a atender aos *stakeholders* em suas diferentes necessidades.

O MEEP foi baseado no *plugin* Metrics2 (http://metrics2.sourceforge.net/), do qual foi reutilizada a parte da arquitetura responsável pela leitura das métricas a serem utilizadas e a extração das métricas propriamente ditas. Este *plugin* foi utilizado por ser de fácil reutilização, ser de código aberto e permitir a inclusão de novas métricas, além de disponibilizar um grande número de métricas voltadas para reutilização (e.g., falta de coesão, profundidade da árvore de herança e número de filhos).

Tanto as informações referentes às extrações quanto os planos de avaliação são armazenados em um banco de dados, que serve como ponte entre a MEEP e a Viz, compartilhando os dados necessários para as visualizações do ReuseDashboard.

3.2. Visualização

Visando atender aos diferentes *stakeholders* de reutilização, o ReuseDashboard possui visões customizáveis, que permitem selecionar quais pares métricas-gráficos devem ser exibidos para cada *stakeholder*. Além disso, para tornar o ambiente apropriado para cada *stakeholder*, o mecanismo pode ser utilizado tanto em plataformas *desktop* quanto em dispositivos móveis, como por exemplo *tablets*. Para isto, o Viz utiliza duas tecnologias: Java EE, para a parte servidor, e o framework Ext JS (http://www.sencha.com/products/extjs/), para a parte cliente. Este framework foi escolhido por se tratar da tecnologia HTML 5 que possibilita uma boa interatividade com objetos visuais, além de possuir vários tipos de gráficos, que são utilizados na parte visual do ReuseDashboard. A Figura 3 ilustra as telas de visualização do ReuseDashboard. A análise realizada por meio do plano de avaliação é exibida na parte inferior da tela, apontando itens nos quais foram detectados problemas.

4. Cenário de Exemplo

Nesta seção, são abordados dois cenários de utilização do ReuseDashboard para exemplificar um dos possíveis benefícios deste mecanismo.

No primeiro cenário, o desenvolvedor de um sistema online precisa melhorar o tempo de processamento. Para isto, este desenvolvedor decidiu investigar a complexidade ciclomática das classes e métodos do componente reutilizado, a fim de encontrar possíveis casos de alta complexidade. Desta forma, o desenvolvedor utiliza o plano de avaliação *Plan A*, exibido na Figura 2, em conjunto com as visualizações fornecidas pelo ReuseDashboard. Assim, ele pode constatar alguns casos cuja complexidade ciclomática era muito alta, além de ter a informação dos itens que ficaram fora dos padrões definidos no plano de avaliação. Estes fatos são evidenciados na Figura 3 (parte superior).

No segundo cenário, um gerente está buscando algum indicativo sobre o esforço de efetuar uma extensão das funcionalidades de um componente. Para isso, este gerente analisa métricas que informem a flexibilidade do código deste componente. A partir das métricas disponíveis no plano de avaliação, ele observa que o nível de abstração (*abstractness*) e o nível de instabilidade (*instability*) podem ser indicativos relevantes. A Figura 3 (parte inferior) ilustra as visualizações destas métricas, que podem ajudar o

gerente nesta tarefa. É possível notar que os valores estão próximos de zero, podendo indicar (dependendo do cenário do projeto) que o código do componente possui a flexibilidade desejada, isto é, é passível de alterações com mais facilidade.

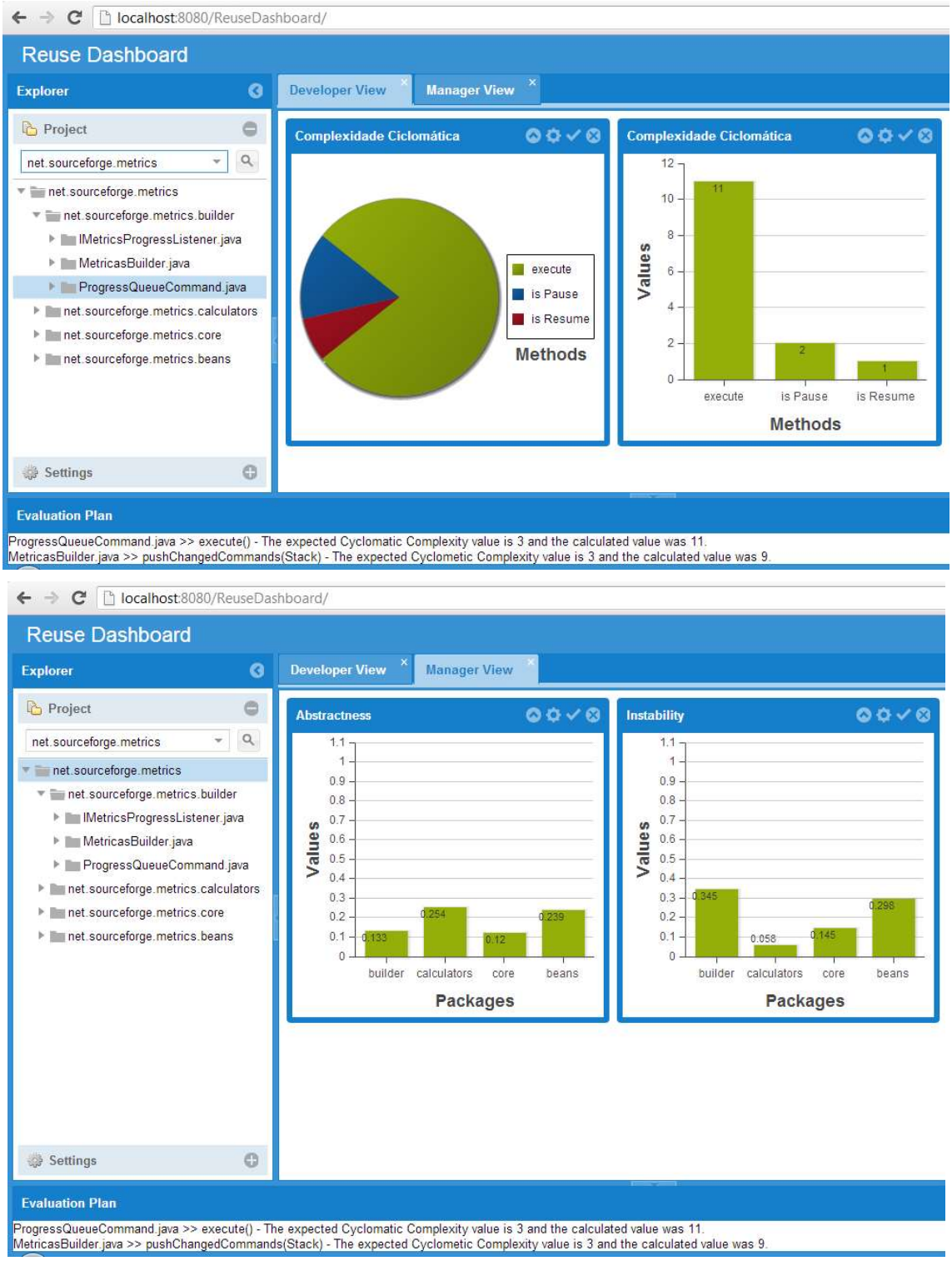

Figura 3 – Visão do Desenvolvedor (parte superior) e do Gerente (parte inferior)

5. Considerações Finais

Este trabalho apresentou o mecanismo ReuseDashboard, que visa apoiar os diversos *stakeholders* presentes no processo de desenvolvimento do software no acompanhamento de um programa de reutilização. O mecanismo disponibiliza informações analíticas em um *dashboard*, que pode ser consultado em plataformas móveis e *desktop*, sendo que as visualizações podem ser parametrizadas para cada *stakeholders* conforme suas necessidades. As informações visualizadas são baseadas em métricas relacionadas à reutilização, extraídas de projetos Java.

Embora a apresentação da visualização seja compatível com plataformas móveis e *desktop*, a interatividade atualmente fica comprometida em dispositivos móveis, em função do framework atualmente utilizado. Para tratar esta limitação, a camada de visualização incorporará os recursos providos pelo framework Sencha Touch (http://www.sencha.com/products/touch), que é compatível com as demais camadas do módulo Vis e permite portar aplicações web escritas em Ext JS para plataformas móveis. Outra limitação relacionada à visualização é a variedade de gráficos, que atualmente é limitada às que são providas pelo framework.

Como próximos passos, pretende-se efetuar a inclusão de novas métricas relacionadas ao reúso (sendo adquiridas a partir de uma revisão sistemática) e efetuar a integração do mecanismo com o ambiente APPRAiSER, em desenvolvimento no contexto de uma tese de doutorado da COPPE/UFRJ, que visa apoiar a reutilização através da percepção por meio de técnicas e recursos de visualização de software [Schots *et al.*, 2012].

Pretende-se, ainda, executar uma avaliação do ReuseDashboard por meio de um experimento, com o objetivo de avaliar a efetividade no acompanhamento do programa de reutilização de software. Serão considerados itens como a facilidade na interpretação das informações visuais, o tempo gasto na análise, e o índice de acertos (baseado num gabarito pré-determinado). Para isto, deverá ser realizada a definição de um plano de avaliação e, a partir do mesmo, a análise e a monitoração, envolvendo tarefas de identificação de componentes, avaliação da qualidade do software, ou identificação de problemas de implementação relacionados à reutilização.

Agradecimentos

Ao Zaedy Sayão, pelo suporte à implementação, e ao CNPq, pelo apoio financeiro.

Referências

Caldiera, G., e Basili, V. R., (1991), “Identifying and qualifying reusable software components”. Computer, vol. 24, no. 2, pp. 61-70, Fevereiro.

Cho, E. S., Kim, M. S., Kim, S. D., (2001), “Component metrics to measure component quality”. In APSEC, IEEE Computer Society, pp. 419-426, Dezembro.

Diehl, S, (2007). “Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software”, 1 ed. Springer

Frakes, W. B., & Isoda, S., (1994), “Success factors of systematic reuse”, IEEE Software, vol 11, no. 5, pp. 15-19, Setembro.

Frakes, W.B., Kang, K., (2005), “Software reuse research: Status and future”, IEEE Transactions on Software Engineering, v. 31, n. 2, pp. 529-536, Julho.

Keim, D. A., Mansmann, F., Schneidewind, J., Thomas, J., & Ziegler, H. (2008). *Visual analytics: Scope and challenges*, pp. 76-90. Springer Berlin Heidelberg.

Kim, Y. and Stohr, E. A. (1998), “Software Reuse: Survey and Research Directions”. Journal of Management Information Systems, Vol. 14, Issue 4, pp. 113-147, March.

Kuipers, T., Visser, J., Vries, G., (2007), "Monitoring the Quality of Outsourced Software", Workshop on Tools for Managing Globally Distributed Software Development (TOMAG), Agosto.

Lanza, M., Marinescu, R. (2006) “Object-Oriented Metrics in Practice”. Springer-Verlag Berlin Heidelberg New York, 1st edition.

Palmieri, M., Werner, C., (2012), “ReuseInvestigator: Um Mecanismo de Identificação de Componentes Reutilizáveis com o Apoio de Visualizações”, II Workshop de Teses e Dissertações do CBSoft (WTDSoft), Natal, Setembro, pp. 1-5.

Plösch, R., Gruber, H., Pomberger, G., Saft, M., Schiffer, S., (2008), “Tool Support for Expert-Centred Code Assessments”, 1st International Conference on Software Testing, Verification, and Validation, pp. 258-267, Lillehammer, Abril.

Schots, M., Werner, C., Mendonça, M., (2012). “Awareness and Comprehension in Software/Systems Engineering Practice and Education: Trends and Research Directions”, 26th Brazilian Symposium on Software Engineering (SBES), Natal, Setembro.

SourceMiner, (2013), Disponível em http://www.sourceminer.org/index.html.

Thomas, J. J., and Cook, K. A. (2006). “A visual analytics agenda”. *IEEE Computer Graphics and Applications*, v. 26, n. 1, pp. 10-13.