Visualização de Software como Suporte ao Desenvolvimento Centrado em Métricas Orientadas a Objetos

Elidiane Pereira dos Santos e Sandro Santos Andrade

Grupo de Sistemas Distribuídos, Otimização, Redes e Tempo-Real (GSORT)
Instituto Federal de Educação, Ciência e Tecnologia da Bahia (IFBa)
Av. Araújo Pinho, nº 39, Canela – 40.110 -150 – Salvador – BA – Brasil
{elidiane, sandroandrade}@ifba.edu.br

Abstract. *Software visualization approaches, mostly those ones directly integrated into development environments, have increasingly been considered major tools when designing and evolving complex and large-scale software systems. This paper reports on the experience of implementing the Lorenz & Kidd suite of object-oriented metrics, as well as three accompanying case-studies for visualizing such metrics. Furthermore, an empirical evaluation has been undertaken, which aimed at assessing the impact of the proposed tool on the quality of artifacts yielded from a code refactoring activity.*

Resumo. *Mecanismos para visualização de software, preferencialmente aqueles integrados em ambientes de desenvolvimento, representam ferramentas cada vez mais importantes para o desenvolvimento e evolução de sistemas computacionais modernos. Este artigo reporta a experiência de implementação da Suite de Métricas OO de Lorenz & Kidd no Ambiente Integrado de Desenvolvimento Qt Creator, a realização de três estudos de caso com a ferramenta desenvolvida e a condução de um experimento com o objetivo de investigar o impacto do uso da visualização de tais métricas na qualidade dos artefatos gerados por uma atividade de refatoração de código.*

1. Introdução

À medida em que os sistemas computacionais se tornam mais complexos, heterogêneos e de larga-escala, a utilização de mecanismos para visualização de *software* [Caserta and Zendra 2011] se torna fator importante no desenvolvimento, evolução e compreensão de tais aplicações. A Orientação a Objetos (OO), enquanto técnica de projeto e programação de sistemas, ainda representa uma das abordagens mais promissoras para gerenciamento de complexidade e suporte à evolução facilitada de aplicações com grandes *codebases*. Dessa forma, a prospecção e utilização de métricas baseadas nos constructos da OO [Lanza and Marinescu 2010] constitui um dos meios mais simples e diretos de suporte à visualização de sistemas desenvolvidos com o uso deste paradigma.

Técnicas de Visualização de Software têm como objetivo a geração de representações visuais que facilitam a compreensão de diferentes aspectos de um sistema de *software* em diferentes granularidades (níveis de abstração). Possíveis aspectos a serem visualizados incluem informações estruturais, comportamentais ou de evolução do *software*. Tais aspectos podem ser investigados em diferentes níveis de abstração, por exemplo com foco em linhas de código, classes ou no âmbito arquitetural [Caserta and Zendra 2011].

Desde a proposta da *Suite CK* [Chidamber and Kemerer 1994], em 1994, uma série de outros modelos para avaliação de qualidade de projeto OO foram apresentados [Sarkar et al. 2008, Purao and Vaishnavi 2003]. Adicionalmente, estudos como [Olague et al. 2007] tentam relacionar empiricamente o uso de tais métricas à predição de determinados atributos de qualidade de *software*, como por exemplo susceptibilidade a falhas.

Este artigo reporta uma experiência de implementação de uma variante das métricas propostas em [Lorenz and Kidd 1994] e sua visualização em um Ambiente Integrado de Desenvolvimento (IDE - *Integrated Development Environment*). Adicionalmente, foram também realizados três estudos de caso e um experimento de avaliação do impacto do uso deste recurso de visualização na qualidade dos produtos resultantes de uma atividade de refatoração de código.

O restante deste artigo está organizado como segue. A seção 2 apresenta as principais métricas presentes em [Lorenz and Kidd 1994]. Na seção 3, os aspectos técnicos e de projeto da implementação realizada são apresentados, enquanto a seção 4 apresenta os estudos de caso e o experimento de avaliação realizados. A seção 5 aponta os trabalhos correlatos e, finalmente, a seção 6 apresenta as conclusões e possíveis trabalhos futuros.

2. A Suite de Métricas OO de Lorenz & Kidd

Lorenz & Kidd, baseados nas métricas presentes na *Suite CK*, propuseram em 1994 um extenso conjunto de cerca de 30 métricas para avaliação de complexidade em sistemas OO. Tais métricas foram classificadas em grupos relacionados a cinco aspectos distintos:

1. Tamanho de Método: inclui métricas tais como quantidade de mensagens enviadas, quantidade de sentenças constituintes, quantidade de linhas de código e tamanho médio de método.
2. Aspectos Internos de Método: o objetivo é estimar a complexidade de manutenção de um método, por exemplo através do cálculo da Complexidade Ciclomática.
3. Tamanho de Classe: inclui métricas tais como quantidade de métodos públicos por classe, quantidade média de métodos por classe, quantidade de atributos por classe, quantidade de atributos/métodos estáticos por classe, etc.
4. Herança de Classe: dentre as métricas deste grupo destacam-se grau de aninhamento dentro de uma hierarquia, quantidade de classes abstratas e uso de herança múltipla.
5. Herança de Método: grupo com o maior número de métricas, incluindo quantidade de métodos sobrepostos por uma sub-classe, quantidade de métodos herdados por uma sub-classe, índice de especialização, coesão da classe, quantidade média de parâmetros por método, uso de funções *friend*, acoplamento entre classes e quantidade de vezes que uma classe é reutilizada.

As seguintes métricas foram implementadas na ferramenta descrita neste trabalho:

1. Relação Média entre Atributos e Métodos de uma Classe (MAC): é obtida pelo total de atributos dividido pelo total de métodos de uma determinada classe [Lorenz and Kidd 1994]. A presença de muitos atributos e poucos métodos é um indício de uma provável baixa coesão na classe em questão.

2. Quantidade de Métodos por Classe (QMC): considera métodos com qualquer visibilidade em uma determinada classe. Segundo [Lorenz and Kidd 1994], o valor recomendado para esta métrica é de 40 para classes de interface gráfica de usuário e 20 para as demais.
3. Tamanho Médio dos Métodos de uma Classe (TMC): considera o número de linhas físicas de código ativo presentes em um método [Lorenz and Kidd 1994]. Métodos de fácil manutenção geralmente possuem tamanho reduzido. O valor estimado é de cerca de 30 linhas para códigos escritos em C++ [Ambler 1998].
4. Quantidade de Atributos por Classe (QAC): é um possível indicador indireto da qualidade do projeto [Lorenz and Kidd 1994]. Uma classe com um alto número de atributos possivelmente sugere a presença de um alto número de relacionamentos com outros objetos do sistema. As classes com mais de três ou quatro atributos mascaram o problema de acoplamento da aplicação [Ambler 1998]. Em classes de interface gráfica de usuário o número de atributos pode chegar a nove, pois estas classes necessitam de mais recursos para lidar com aspectos de interação [Lorenz and Kidd 1994].
5. Quantidade de Métodos Públicos (QMP): visto que os métodos públicos representam aqueles serviços disponíveis para objetos de outras classes, esta métrica geralmente constitui um bom indicador do grau de coesão da aplicação.

3. A Ferramenta Proposta

O *Qt Creator Visualization Plugin*, ferramenta apresentada neste artigo, implementa a extração e visualização das métricas acima descritas, de forma integrada ao *Qt Creator* [Qt-Project 2013] - IDE padrão para desenvolvimento Qt. O Qt é atualmente um dos *toolkits* para desenvolvimento multi-plataforma mais completos e promissores. Desde o início do seu desenvolvimento, em 1991, o Qt vem se consolidando não só como um poderoso *toolkit* para construção de interfaces graficas de usuário mas também como excelente biblioteca de propósito geral, com recursos por exemplo para *Inter-Process Communication*, acesso a banco de dados, programação concorrente e distribuída, manipulação de XML, renderização 3D, dentre outras funcionalidades.

O *Qt Creator* disponibiliza os principais recursos para construção facilitada de interfaces gráficas de usuário, depuração, *syntax highlighting*, funções básicas de refatoração, *profiling* e integração com sistemas de controle de versão. Entretanto, é ainda carente de recursos para visualização de *software*, apresentando apenas grafos simples gerados pela ferramenta de *profiling* Valgrind. Este trabalho projetou e implementou um *plugin* para visualização das métricas de *software* descritas, de forma integrada ao *Qt Creator*.

O principal requisito do *plugin* é calcular e visualizar as métricas por classe de um determinado projeto. Para isso, o *plugin* realiza a extração dos dados a partir da infraestrutura interna de representação de código-fonte do *Qt Creator* e calcula as métricas desejadas. A ferramenta exibe graficamente os valores das métricas, possibilitando a avaliação da qualidade do programa desenvolvido, e sinaliza valores limite que uma determinada métrica pode assumir. Isso possibilita a rápida identificação das classes e métodos mais problemáticos do projeto. A ferramenta foi desenvolvida em C++ no ambiente *Qt Creator* e requer somente a instalação do *plugin* correspondente para a sua utilização imediata.

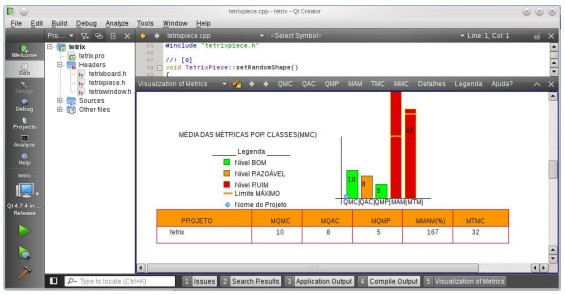

Figura 1. Projeto Tetrix - resumo das métricas

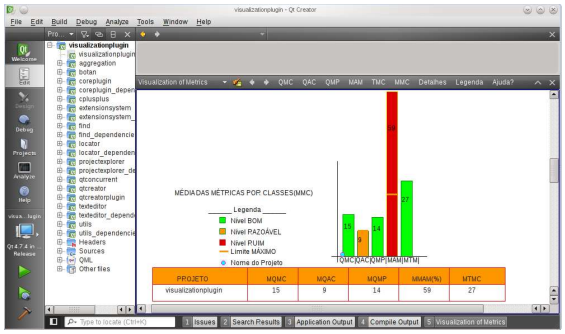

Figura 2. Visualização do próprio plugin - resumo das métricas

4. Avaliação

A ferramenta foi avaliada em duas dimensões: capacidade de extração das métricas em projetos de médio/grande porte (teste funcional) e análise do impacto de uso da ferramenta na qualidade dos artefatos gerados por operações de refatoração. As seções seguintes apresentam os estudos de caso utilizados no teste funcional e o experimento realizado para avaliação do impacto de uso da ferramenta.

4.1. Estudo de Caso 1: Tetrix

O primeiro estudo de caso analisou a utilização da ferramenta em um projeto de pequeno porte: o Tetrix. O jogo é formado por três classes: `TetrixWindow`, `TetrixBoard` e `TetrixPiece`. A classe `TetrixBoard` é a classe mais complexa pois lida com a lógica do jogo e a renderização. A Figura 1 exibe a visão geral da qualidade do sistema com o resumo das métricas.

A métrica QMC apresentou nível bom, pois obteve o valor de 10 métodos por classe, a métrica QAC nível razoável de 8 atributos por classe, enquanto a métrica QMP um nível bom de 5 métodos públicos por classe. A métrica MAC apresentou nível ruim (167%), ultrapassando o valor estimado de 22% de atributos por métodos. Solucionar este problema requer que a quantidade de atributos por método seja reduzida até que o *plugin* indique a adequação à faixa estimada. A métrica TMC apresentou nível ruim (32 linhas), pois o seu limite é de 30 linhas de código por métodos.

4.2. Estudo de Caso 2: *Qt Creator Visualization Plugin*

O segundo estudo de caso analisou o código da própria ferramenta implementada: um projeto de pequeno porte com apenas três classes, porém complexas por ter que extrair dados do código-fonte e ser uma extensão de uma IDE. O projeto é composto pelas classes: `VisualizationMetricsPlugin`, `VisualizationMetricsPane` e `VisualizationMetricsCollector`. A Figura 2 apresenta os resultados para este estudo de caso.

4.3. Estudo de Caso 3: *Qt Creator*

O terceiro estudo de caso analisou o *Qt Creator*: projeto de grande porte composto por mais de 3000 classes. A Figura 3 exibe parte da visualização das métricas. A Figura 4 apresenta o resumo das métricas. Para calcular as métricas deste projeto o *plugin* levou em torno de 10 segundos de execução.

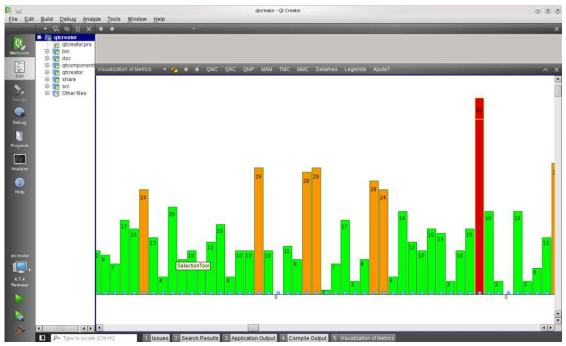

Figura 3. Projeto Qt Creator - métrica QMC

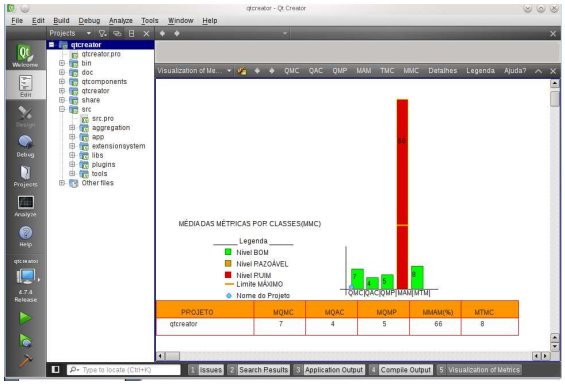

Figura 4. Projeto Qt Creator - resumo das métricas

4.4. Avaliação Experimental

Para estimar o impacto da utilização do *plugin* em atividades de refatoração de aplicações desenvolvidas em C++/Qt, um experimento com oito participantes foi conduzido com o objetivo de avaliar as seguintes hipóteses:

- Hipotese Nula (H_0): a utilização do *plugin* de visualização de *software* não implica em uma diferença estatisticamente considerável na diferença média das métricas antes e depois da refatoração.
- Hipotese Alternativa (H_1): a utilização do *plugin* de visualização de *software* implica em uma diferença estatisticamente considerável na diferença média das métricas antes e depois da refatoração.

Espera-se que, se rejeitada a hipótese nula, o valor das métricas após a refatoração utilizando o *plugin* indique uma melhoria na qualidade interna do sistema. Ou seja, se uma métrica menor indicar um sistema de melhor qualidade espera-se que as métricas após refatoração com o uso do *plugin* sejam menores que as métricas após refatoração sem o uso do *plugin*.

Para isso foi conduzido um experimento onde o jogo de Tetrix deveria ser refatorado. Neste experimento, o objetivo foi realizar operações de refatoração que melhorassem aspectos tais como manutenibilidade e legibilidade do codigo-fonte. Foi definido um grupo de controle, que não utilizou o *plugin*, e outro grupo que realizou as operações com o auxílio da ferramenta. Todas as métricas foram medidas antes e após o experimento para cada indivíduo de cada um dos dois grupos apresentados. O experimento possui um fator (uso do *plugin*), com dois níveis (sim ou não), sendo executado de forma emparelhada (os dois grupos foram sujeitos ao mesmo conjunto de atividades realizadas).

A Figura 5 apresenta o gráfico consolidado da métrica QMC para todo o projeto. Contém, para cada indivíduo, as médias das métricas obtidas de todas as classes do projeto em comparação com as métricas originais. Os indivíduos 1 e 2 retiraram a quantidade de métodos em excesso das classes, deixando-as com um valor aceitável para a métrica QMC. O indivíduo 3 manteve o *codebase* estável, não acrescentou nem retirou nenhum método das classes. O indivíduo 4 adicionou mais métodos às classes, porém a métrica continuou na faixa aceitável. No gráfico do "Grupo de Uso do Plugin", os indivíduos 1, 2 e 3 acrescentaram métodos às classes porém não ultrapassaram o limite, adicionando mais funcionalidades ao sistema. O indivíduo 4 manteve o *codebase* estável.

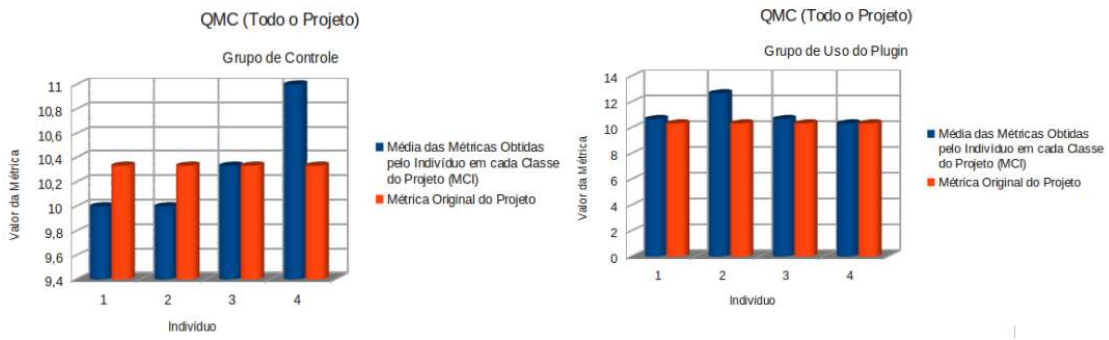

Figura 5. Gráficos consolidados para a métrica QMC em cada grupo

4.4.1. Análise dos Resultados

O teste estatístico *t-test* foi utilizado para avaliar se a diferença entre as médias das duas amostras é estatisticamente significante. Realizou-se o *t-test* para cada métrica sendo avaliada, assumindo que as duas amostras (resultados sem o uso do *plugin* e resultados com o uso do *plugin*) foram obtidas de duas populações com distribuição normal, médias desconhecidas e variâncias desconhecidas porém iguais.

Deseja-se conhecer o maior nível de confiança para o qual o *plugin* produz uma diferença estatisticamente significante na média das métricas:

- *t-test* para a métrica QMC: *p-value* = 0,3618456438; Nível máximo de confiança (1-*p-value*) = 0,6381543562;
- *t-test* para a métrica QAC: *p-value* = 0,391002219; Nível máximo de confiança (1-*p-value*) = 0,608997781;
- *t-test* para a métrica QMP: *p-value* = 0,8542703292; Nível máximo de confiança (1-*p-value*) = 0,1457296708;
- *t-test* para a metrica MAC: *p-value* = 0,8354105574; Nível máximo de confiança (1-*p-value*) = 0,1645894426;
- *t-test* para a métrica TMC: *p-value* = 0,2622190236; Nível máximo de confiança (1-*p-value*) = 0,7377809764;

Pode-se afirmar, com base nos resultados, que a métrica mais influenciada pelo uso do *plugin* foi a TMC, já que existe 73% de certeza que o *plugin* faz diferença na diferença desta métrica antes e depois das operações de refatoração. Outras métricas potencialmente afetadas pelo *plugin* foram a QMC (63%) e QAC (60%). As demais possuem nível de confiança extremamente baixo e nada pode-se afirmar.

5. Trabalhos Correlatos

Dentre as ferramentas relacionadas à proposta apresentada neste artigo destacam-se: *Visual Metrics* [Rosenberg 1998], ferramenta para cálculo de métricas de Seibt [da Silva Seibt 2001] e a visualização de métricas para acoplamento e coesão de [Dantas 2008].

O *Visual Metrics* permite o cálculo de algumas métricas de construção e de projeto, possibilitando a análise de códigos-fonte desenvolvidos em C# e Java. Com o *Visual*

Tabela 1. Comparação entre as ferramentas de métricas OO

Ferramentas	Métricas					Linguagens			Gráficos	IDE
	MAC %	QMC	TMC	QMP	QAC	Java	C++	Delphi		
Qt Creator Visualization Plugin	✔	✔	✔	✔	✔		✔		✔	✔
Visual Metrics						✔			✔	
Protótipo de Seibt	✔	✔	✔	✔	✔			✔		
Ferramenta de Marques						✔			✔	

Metrics o desenvolvedor pode selecionar os arquivos de projeto a serem analisados, selecionar elementos individuais para análise específica, escolher as métricas a serem calculadas, armazenar e recuperar métricas calculadas e definir limites mínimo e máximo que uma determinada métrica poderá atingir.

A ferramenta para cálculo de métricas de Seibt é um protótipo que fornece métricas pré-definidas a partir da análise de códigos-fonte de programas desenvolvidos em Delphi. As métricas selecionadas foram as de projeto e construção totalizando 19 métricas diferentes.

A visualização de Marques utiliza métricas para indicar o grau de coesão e acoplamento entre classes, exibindo pontos do projeto que pareçam menos estruturados ou pior implementados. A ferramenta analisa o código-fonte de um projeto Java, indicando o grau de coesão e acoplamento nestes sistemas. As métricas calculadas por esta ferramenta são as seguintes métricas da *Suite CK*: CBO (*Coupling Between Objects*), DIT (*Depth of Inheritance Tree*), NOC (*Number of Children*) e LCOM3 (*Lack of Cohesion of Methods*).

A ferramenta apresentada neste artigo se difere das demais nos seguintes aspectos:

- Disponibiliza representação gráfica dos valores calculados, diferente por exemplo da ferramenta de Seibt;
- É integrada a uma IDE, dispensando a utilização de outras ferramentas para compreensão e análise do projeto sendo desenvolvido;
- Implementa algumas métricas de construção, fundamentais e importantes, que o *Visual Metrics* e a ferramenta de Marques não implementam;
- Utiliza uma representação visual diferente para cada métrica, apresentando detalhes de cada uma. Nas demais soluções apenas um gráfico é utilizado para todas as métricas.

A Tabela 1 apresenta uma comparação entre as ferramentas apresentadas.

6. Conclusões e Trabalhos Futuros

Este artigo apresentou uma visão geral sobre o projeto e desenvolvimento de um *plugin* para a IDE *Qt Creator* que extrai e visualiza algumas das métricas descritas na *suite* de Lorenz & Kidd. A ferramenta calcula cinco métricas fundamentais para avaliar a qualidade de um *software*. Com a análise dos estudos de casos e experimento realizados, nota-se a importância da utilização de ferramentas para extração e visualização de métricas na melhoria da qualidade do produto final, desde sistemas simples com poucas classes até aplicações de grande porte.

Dentre as limitações da proposta apresentada pode-se destacar a forma incipiente e com escalabilidade limitada do paradigma de visualização utilizado (gráficos simples), se comparado a técnicas mais sofisticadas, tais como *tree maps* ou *hierarchical radial bundles*. Trabalhos futuros incluem o projeto de experimentos mais robustos e com melhor controle, implementação de métricas de âmbito arquitetural e recursos para integração/navegação entre visualização e edição de código-fonte.

Referências

Ambler, S. W. (1998). *Building Object Applications That Work: Your Step-by-Step Handbook for Developing Robust Systems with Object Technology*. Cambridge University Press.

Caserta, P. and Zendra, O. (2011). Visualization of the Static aspects of Software: a survey. *IEEE Transactions on Visualization and Computer Graphics*, 17(7):913–933.

Chidamber, S. R. and Kemerer, C. F. (1994). A metrics suite for object oriented design. *IEEE Trans. Softw. Eng.*, 20(6):476–493.

da Silva Seibt, P. R. R. (2001). Ferramenta para cálculo de métricas em softwares orientados a objetos. In *Trabalho de Conclusão de Curso - Graduação em Ciência da Computação - Universidade Regional de Blumenau.*

Dantas, M. M. M. (2008). Métricas para acoplamento e coesão em sistemas orientado a objetos em ambiente de visualização de software. In *Trabalho de Conclusão de Curso - Graduação em Ciência da Computação - Universidade Federal da Bahia (UFBa).*

Lanza, M. and Marinescu, R. (2010). *Object-Oriented Metrics in Practice. Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-oriented Systems*. Springer Verlag.

Lorenz, M. and Kidd, J. (1994). *Object-oriented software metrics: a practical guide*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Olague, H. M., Etzkorn, L. H., Gholston, S., and Quattlebaum, S. (2007). Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes. *IEEE Trans. Softw. Eng.*, 33(6):402–419.

Purao, S. and Vaishnavi, V. (2003). Product metrics for object-oriented systems. *ACM Comput. Surv.*, 35(2):191–221.

Qt-Project (2013). Qt creator 2.8 online documentation. *http://qt-project.org/doc/qtcreator-2.8. Acesso: Agosto de 2013.*

Rosenberg, L. H. (1998). Applying and Interpreting Object Oriented Metrics. In *Software Technology Conference*. NASA Software Assurance Technology Center (SACT).

Sarkar, S., Kak, A. C., and Rama, G. M. (2008). Metrics for measuring the quality of modularization of large-scale object-oriented software. *IEEE Trans. Software Eng.*, 34(5):700–720.