ISSN 2236-8884

Uma Ferramenta de Métricas de Modelo para Apoiar a Garantia da Qualidade de Software

A Model Metrics Tool to Support Software Quality Assurance

Mara R. dos S. Barcelos, Michael C. da Silva, Aline P. V. de Vasconcelos

Instituto Federal Fluminense campus Campos Centro
Rua Dr. Siqueira, 273 – Parque Dom Bosco – CEP 28030-130 – Campos dos Goytacazes – RJ – Brasil
{mrbarcelos, mcaldas, apires}@iff.edu.br

Abstract: *This paper presents XMIMetricsTool, a model metrics tool developed to assure software quality from the beginning of its life cycle. The tool is generic and free. It calculates the metrics by reading XMI files (XML Metadata Interchange), generated from the class model within the system. Therefore, faults can be detected early, not evolving to the development phase and, thus, reducing the cost of their correction and maintenance. Moreover, the tool meets some of the expected results in the Measure Process of Mps.Br (Brazilian Software Process Improvement).*

Key words: *XMIMetricsTool. Model metrics tool. Software quality assurance.*

Resumo: Este artigo apresenta a XMIMetricsTool, uma ferramenta de métricas de modelo que foi desenvolvida para assegurar a Garantia de Qualidade de Software desde o início do seu ciclo de vida. A ferramenta é livre e genérica. Calcula as métricas através da leitura de arquivos XMI (*XML Metadata Interchange*) gerados a partir do modelo de classes do sistema. Assim, os defeitos podem ser descobertos cedo, não evoluindo para a fase de implementação, reduzindo o seu custo de correção e manutenção. Além disso, a ferramenta atende alguns dos resultados esperados do Processo de Medição do Mps.Br (Melhoria de Processos do Software Brasileiro).

Palavras Chaves: XMIMetricsTool. Ferramenta de métricas de modelo. Garantia de Qualidade de Software

ISSN 2236-8884

CITI 2010 9 a 11 de novembro
6º Circuito de Tecnologia da Informação

Introdução

As métricas são necessárias para verificar a qualidade do produto e produtividade do processo de desenvolvimento de *software*.

Segundo Pressman (2002), o *software* é medido por várias razões: indicar a qualidade do produto; avaliar a produtividade das pessoas que desenvolvem o produto; avaliar os benefícios em termos de produtividade e qualidade derivados de novos métodos e ferramentas de *software*; formar uma linha básica de estimativas; e, ajudar a justificar os pedidos de novas ferramentas ou treinamento adicional. O processo de desenvolvimento de *software* é uma atividade criativa, que não depende exclusivamente das ciências exatas, mas muito mais dos fatores humanos. Isso ocorre porque o *software* é um produto abstrato e não concreto como em outras áreas da engenharia. Sendo assim, o processo de medição na Engenharia de *Software* é realizado através do uso de métricas, que são escolhidas de acordo com as características do *software* que se quer medir e com os objetivos da medição. A medição de *software* se dedica a apresentar um valor numérico para algum atributo de um produto ou processo de *software*.

Como a qualidade é um fator indispensável, existem muitas atividades com o intuito de garantir a qualidade de um produto de *software*, como inspeções, testes e medição, porém elas acontecem em fases diferentes do ciclo de vida do *software*. As inspeções devem ser realizadas após cada fase do ciclo de vida para detectar falhas. Podem ocorrer desde a análise e projeto, detectando erros no início do ciclo de vida. Porém, costumam ser manuais e, dependendo da técnica utilizada, agregam alguma subjetividade ao processo. Os testes, de caixa branca ou preta, são técnicas bastante utilizadas, mas começam a ser empregadas em fases de desenvolvimento (implementação) e/ou implantação. Dessa forma, detectam defeitos em uma fase avançada do ciclo de vida do *software*, aumentando seu custo de correção. Já as métricas, no caso deste trabalho as métricas de modelo, podem ser utilizadas desde as fases de análise e projeto e representam um valor objetivo para a verificação de pontos críticos do projeto que podem levar a falhas no *software*. Com isso, possíveis erros podem ser identificados de maneira precoce, reduzindo custo com manutenções posteriores.

Sendo assim, a XMIMetricsTool foi desenvolvida com o objetivo de garantir a qualidade do *software* desde o início do seu ciclo de vida. A ferramenta é genérica, pois realiza suas funcionalidades através da leitura de arquivos do tipo XMI, que é comum à maioria das ferramentas *CASE*. XMI é o que

PRO IFF · Essentia Editora · Instituto Federal de Educação, Ciência e Tecnologia Fluminense
Secretaria de Educação Profissional e Tecnológica · Ministério da Educação · Governo Federal Brasil País rico é país sem pobreza

ISSN 2236-8884

se pode dizer de uma maneira simples, um XML usado para a troca de informações entre ferramentas *CASE* que têm por base a linguagem de modelagem UML. Outra finalidade da ferramenta é ser livre (*free*), por este motivo será disponibilizada no portal do *software* público.

Este artigo está divido em mais quatro capítulos, além desta introdução, a saber: o Capítulo 2 descreve o referencial teórico; o Capítulo 3 descreve a MXIMetricsTool, no Capítulo 4, a relação entre o MPS.Br e a XMIMetricsTool é explicada; o Capítulo 5 apresenta os trabalhos relacionados; por fim, o Capítulo 6 apresenta as conclusões e os resultados obtidos.

Referencial Teórico

As métricas originaram-se da aplicação de cálculos para quantificar indicadores sobre o processo de desenvolvimento de um *software*, sendo adotadas a partir de 1970. Existem quatro tendências dessa tecnologia, vista em (MOLLER, 1993): **a)** medida de complexidade do código: técnica desenvolvida em meados de 1970, para avaliar a complexidade do código, através da análise do mesmo; **b)** estimativa do custo de um projeto de *software*: esta técnica foi desenvolvida em meados de 1970, estimando o trabalho e o tempo gasto para se desenvolver um *software*, baseando-se além de outros fatores, na quantidade de linhas de código necessárias para a implementação; **c)** garantia da qualidade do *software*: estas técnicas foram criadas visando implementar qualidade ao produto de *software*, e foram melhoradas significativamente entre os anos de 1970 e 1980; **d)** processo de desenvolvimento do *software*: o projeto de *software* tornou-se grande e mais complexo, com isso, surgiu a necessidade de controlar este processo.

Os desenvolvedores de sistemas, a partir do aparecimento dessas tendências, começaram então a utilizar as métricas com o propósito de melhorar o processo de desenvolvimento do *software* e o produto como um todo.

Métricas de Modelo

Para Rocha (1994), dadas as diferenças entre as visões Orientada a Objetos (OO) e procedural, é comum constatar que as métricas de *software* desenvolvidas para serem aplicadas aos métodos tradicionais de desenvolvimento não são facilmente mapeadas para os conceitos da OO. As métricas

ISSN 2236-8884

para *software* OO são diferentes devido à localização, ao encapsulamento, à herança e técnicas de abstração dos objetos, que reúnem dados e funções. Sendo algumas delas obtidas através do código e outras através do modelo. Métricas de modelo são aquelas que podem ser extraídas desde o modelo, o que não impede que elas sejam extraídas através do código também, porém existem métricas que só podem ser obtidas através da análise do código. Em outras palavras, as métricas de modelo podem ser extraídas a partir do código, mas métricas de código não podem ser extraídas através do modelo, pois só podem ser extraídas na fase de desenvolvimento.

As métricas de Chidamber e Kemerer (1994), conhecidas como conjunto de métricas CK, foram propostas para captar diferentes aspectos de um objeto e de sistemas orientados a objetos, e é o mais amplamente referenciado na literatura e utilizado pela comunidade de Engenharia de Software. No entanto, algumas métricas desse conjunto são obtidas somente a partir do código, como por exemplo Métodos Ponderados por Classe (WMC – *Weighted Methods Per Class*), que é uma contagem dos métodos implementados numa classe e a soma de suas complexidades. Essa métrica indica o número de caminhos independentes que podem ser seguidos dentro de um procedimento.

Lorenz e Kidd (1994) propuseram em seu livro sobre fundamentos de qualidade de *software* um conjunto de métricas baseadas em tamanho, herança, aspectos internos e aspectos externos de classes.

O conjunto de métricas desenvolvido em Abreu, Goulao e Esteves (1995) são métricas de projeto e a ênfase das mesmas está nas classes, em outras palavras, em características das classes, como herança, encapsulamento e acoplamento.

Métricas Implementadas na XMIMetricsTool

As métricas implementadas na XMIMetricsTool, como já mencionado, são métricas de modelo, que têm esse nome justamente porque são obtidas através de diagramas de classes UML dos *softwares* que estão sendo desenvolvidos, ou seja, na fase inicial do seu ciclo de vida. Nesta fase, por meio da leitura do arquivo XMI, é possível extrair métricas sobre o modelo de classes. Porém, como o arquivo contempla outros modelos, é possível futuramente ampliar a capacidade de leitura e detecção da ferramenta.

Foram selecionados 2 conjuntos de métricas de modelo de 4 autores, a saber, Chidamber e

ISSN 2236-8884

Kemerer (1994), Lorenz e Kidd (1994). Os conjuntos de métricas desses autores são os mais utilizados pela comunidade de Engenharia de Software e os mais referenciados na literatura. Algumas métricas de modelo desses autores não foram selecionadas, uma vez que o objetivo é demonstrar a viabilidade da abordagem e ferramenta desenvolvida com um conjunto inicial de métricas para apoiar a Garantia da Qualidade de Software no início do ciclo de vida, onde o custo de correção dos erros é mais baixo. A seguir estão relacionadas as métricas implementadas na ferramenta.

Dos autores Chidamber e Kemerer (1994):

• Profundidade da Árvore de Herança (Depth of Inheritance Tree - DIT)

A profundidade da árvore da herança é o número máximo de passos da classe nó até a raiz da árvore, e é medida pelo número de classes ancestrais. Segundo Ambler (1998), essa métrica indica as dificuldades na maneira de utilizar o conceito de herança. Árvores muito profundas constituem projetos de maior complexidade, uma vez que um número maior de métodos e classes estão envolvidos, tornando-a mais difícil de entender e, portanto, de manter e evoluir.

• Número de Filhos (Number of Children - NOC)

O número de filhos consiste no número de subclasses imediatamente subordinadas a uma classe na hierarquia. O número de filhos é um indicador do potencial de influência que uma classe tem no projeto e no sistema. Um alto número de filhos promove maior reutilização, visto que a herança é uma forma de reutilização. Porém, quanto maior o número de filhos, maior a probabilidade de abstração imprópria da classe pai e poder ser um caso de mau uso da herança (CHIDAMBER; KEMERER, 1994].

Dos autores Lorenz e Kidd (1994):

• Número de Métodos Públicos (Number of Public Methods – NPM)

Esta métrica é uma contagem do número de métodos públicos em uma classe, não incluindo os herdados. É usada para ajudar a estimar o esforço necessário para desenvolver uma classe, pois quanto mais são os pontos de acesso a uma classe, mais complexa ela se torna.

• Número de Métodos (Number of Methods - NM)

Segundo Lorenz e Kidd (1994), essa métrica conta todos os métodos de uma classe, incluindo públicos, privados e protegidos, não incluindo os herdados. Esse número relaciona-se com o total de serviços que a classe oferece, e quanto mais serviços, mais complexa de manter é a classe. Além disso,

ISSN 2236-8884

ela pode perder a sua coesão.

• Número de Atributos Públicos (Number of Public Variables - NPV)

Conforme Lorenz e Kidd (1994), ao contar a quantidade de atributos públicos de uma classe, tem-se um indicador da qualidade do projeto, uma vez que o conceito de encapsulamento é ferido.

• Número de Atributos (Number of Variables - NV)

Essa métrica conta o número total de atributos de uma classe, sendo eles públicos, privados ou protegidos, não incluindo os herdados. Classes que contêm muitos atributos podem indicar que talvez haja necessidade de especialização.

• Tamanho da Classe (Class Size- CS)

Essa métrica conta o tamanho total de uma classe, incluindo o número total de atributos e métodos, sendo eles públicos, privados ou protegidos. Classes muito grandes tendem a perder sua coesão. São as chamadas God Classes.

• Número de Operações Adicionadas na Subclasse (Number of Operations Added by a Subclass - NOA).

Essa métrica está diretamente ligada à abstração do projeto. As subclasses devem definir novos métodos, estendendo o comportamento das superclasses. Sendo assim, uma subclasse que não possui métodos únicos é questionável no sistema.

Os valores de referência do grau de normalidade das métricas são obtidos através da média de valores do próprio modelo que está sendo avaliado, uma vez que não existe um consenso na literatura acerca de "bons" valores para essas métricas. Assim sendo, as empresas podem montar um *baseline* de métricas e definir os valores de referência para as mesmas, de acordo com sua realidade e experiência.

Alguns autores propõem valores de referência (CHIDAMBER; KEREMER, 1994; LORENZ; KIDD, 1994), mas não há estudos experimentais que comprovem que os mesmos são válidos para diferentes tipos de sistemas.

Processo de Medição do MPS.Br

O MPS.Br é um programa de Melhoria do Processo do Software Brasileiro, que define níveis de maturidade que devem ser atingidos pelas empresas (MPS .Br, 2010a). Cada nível possui processos,

ISSN 2236-8884

cada processo tem resultados específicos que devem ser atendidos. Um dos processos definidos pelo MPS.Br, para o nível de maturidade F, é o de Medição, cujo o propósito é coletar, armazenar, analisar e relatar os dados relativos aos produtos desenvolvidos e aos processos implementados na organização e em seus projetos, de forma a apoiar os objetivos organizacionais.

XMIMetricsTool

O objetivo da ferramenta de métricas é garantir a qualidade desde o início do ciclo de vida do *software*. O processo de medição na ferramenta de métricas desenvolvida está dividido em três etapas: a primeira etapa do processo é o planejamento, na qual a equipe seleciona as métricas dentre as disponíveis na ferramenta que serão utilizadas; a segunda é a extração dos valores das métricas; e análise dos resultados obtidos. Para a tomada de decisão da equipe, um método como o GQM (*Goal--Question-Metric*) (GQM, 2010) pode ser utilizado.

O processo se dá da seguinte maneira: um arquivo XMI é gerado a partir de uma ferramenta CASE utilizada para modelagem do sistema; a seguir, a XMIMetricsTool importa esse arquivo XMI e faz a leitura do mesmo; o usuário deve selecionar as métricas que estão separadas por autores; o próximo passo é solicitar o cálculo dessas métricas e verificar os resultados, através de tabelas com informações sobre as métricas, destacando quando houver uma discrepância de valores acima da média para as métricas calculadas através da cor vermelha no texto; sendo assim o usuário pode analisar os resultados e tomar decisões.

Para exemplificar a utilização da ferramenta desenvolvida, foi utilizado o modelo hipotético de um sistema escolar, que contém os dados necessários para testar a ferramenta, como herança, classes especializadas, atributos e métodos (públicos, privados e protegidos). O estudo de caso do sistema escolar hipotético se desenvolve da seguinte maneira: os alunos se matriculam em cursos, em que cada aluno pode se matricular em cursos superiores, cursos técnicos ou cursos de pós-graduação. Os cursos de pós-graduação possuem área temática, e cursos técnicos são divididos em módulos. A Figura 1 mostra uma parte do diagrama de classes desse modelo.

ISSN 2236-8884

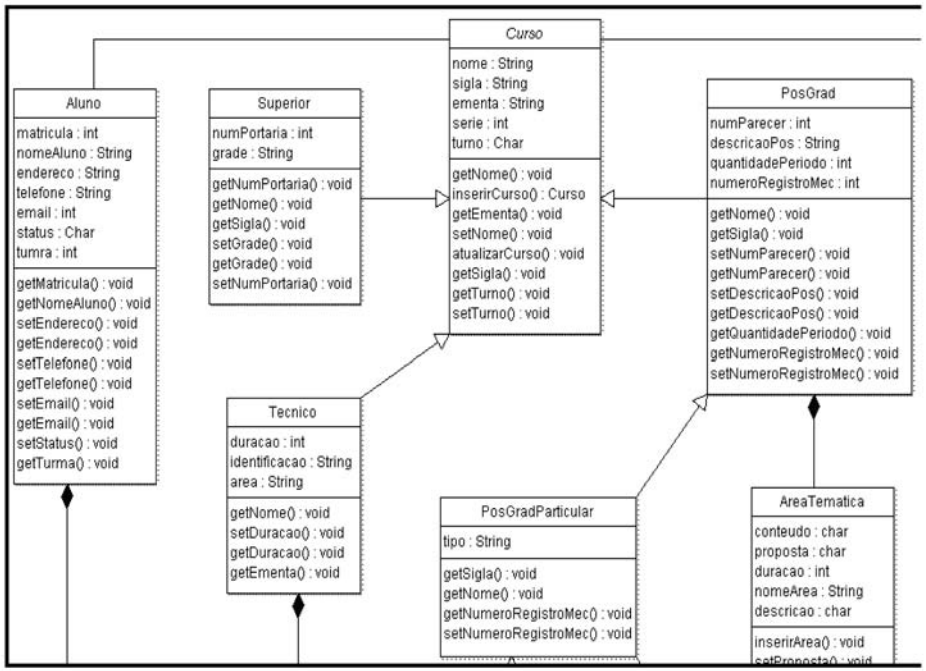

Figura 1: Trecho do Diagrama de Classe do Sistema Escolar Hipotético

Planejamento, Cálculo e Análise das Métricas

No seu estágio atual, a XMIMetricsTool lê arquivos XMI gerados a partir da ferramenta *CASE* ArgoUml. O primeiro passo para dar início ao cálculo das métricas é abrir um arquivo do tipo XMI. Para fazer isso, o usuário deve selecionar a opção Arquivo, na tela inicial. Feito isso aparecerá a opção Abrir um arquivo XMI.

O Sistema possui um *menu* que separa as métricas por seus autores. Primeiramente o usuário seleciona o autor desejado, e depois a métrica que será calculada, como mostra a Figura 2.

ISSN 2236-8884

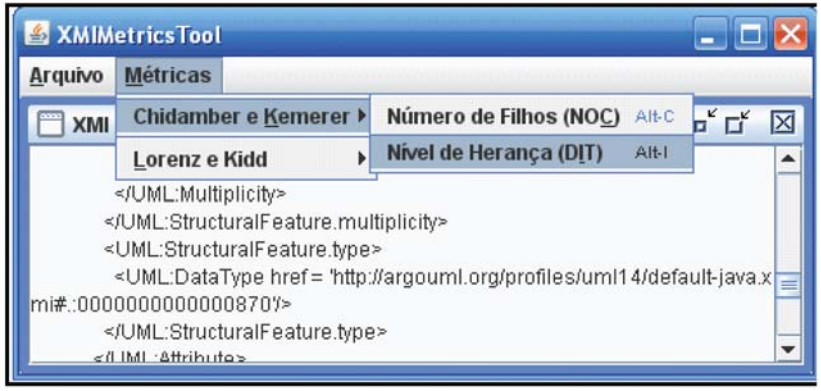

Figura 2: *Tela de Seleção de Autores e Métricas*

A Figura 3 demonstra o resultado do cálculo da métrica NM (*Number of Methods*) dos autores Lorenz e Kidd (1994).

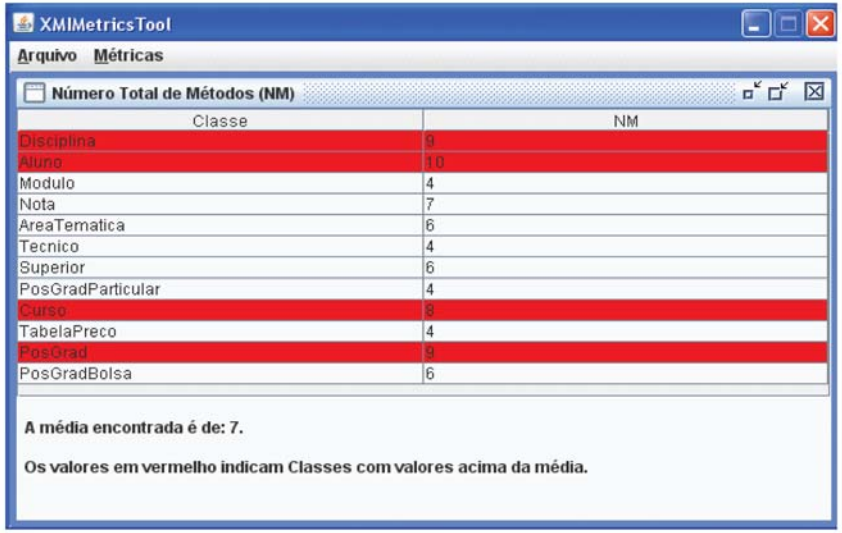

Figura 3: *Cálculo da Métrica NM*

ISSN 2236-8884

As classes Disciplina, Aluno, Curso e PosGrad estão destacadas em vermelho, pois os valores encontrados para a métrica em questão estão acima da média do modelo avaliado.

Processo de Medição do Mps.Br e a XMIMetricsTool

O propósito do processo Medição é coletar, armazenar, analisar e relatar os dados relativos aos produtos desenvolvidos e aos processos implementados na organização e em seus projetos, de forma a apoiar os objetivos organizacionais (MPS Br, 2010b).

O processo de Medição é um processo que apoia os processos de gerência e melhoria de processo e de produto. A seguir estão descritos os resultados esperados do Processo de Medição que são atendidos pela ferramenta apresentada nesse trabalho.

• Resultado MED 5. Os Dados Requeridos são Coletados e Analisados

Os dados devem ser coletados de acordo com o procedimento de coleta estabelecido. Depois de coletados, os dados devem ser analisados conforme o planejado pelas pessoas que têm essa responsabilidade dentro da organização (MPS Br, 2010b). Um dos objetivos da ferramenta é apoiar a tomada de decisões, o que ocorre após o cálculo de uma determinada métrica e a apresentação dos seus resultados, destacando-se os valores acima da média.

• Resultado MED 7. Os Dados e Resultados da Análise são Comunicados e Utilizados para Tomar Decisões

As informações produzidas devem ser comunicadas aos usuários das medições, apoiando-os nos processos de tomada de decisão (MPS Br, 2010b).

Os resultados obtidos com os cálculos das métricas devem ser comunicados a toda equipe, para que possam ser analisados e discutidos, apoiando posterior tomada de decisão. Isso é possível porque a ferramenta calcula a média de valores e mostra os pontos fora da curva, ou seja, os valores acima da média que devem ser analisados com cuidado.

ISSN 2236-8884

Trabalhos Relacionados

Algumas ferramentas para cálculo de métricas foram testadas, com objetivo de expor suas características principais e identificar pontos em aberto, em que a ferramenta proposta neste trabalho poderia atuar.

O SDMetrics (SDMetrics 2010) é uma ferramenta que suporta a extração de métricas, inclusive as utilizadas na ferramenta exposta nesse trabalho, sobre diversos diagramas UML, porém não é livre.

O Objecteering (Objecteering 2010) é uma ferramenta de modelagem que pode ser usada para coletar métricas em modelos de classe UML. Porém, os modelos devem ser construídos na própria ferramenta, ou seja, não é uma ferramenta genérica. Essa ferramenta também não é livre.

O Together (Borland 2010) é um ambiente de modelagem UML que permite o desenvolvimento de *software* OO, desde a especificação de requisitos até o projeto de casos de teste. Possui um módulo de garantia de qualidade que é o responsável pelo cálculo das métricas. Um dos destaques é que possui um abrangente conjunto de métricas fixas, que incluem as métricas dos conjuntos de Chidamber e Kemerer (1994) e de Lorenz e Kiddd (1994), entre outras. O diagrama precisa ser desenvolvido na própria ferramenta para o cálculo das métricas, ou seja, ela não é genérica, e também não é livre.

O JDepend (JDepend 2010) é um *plug-in* do Eclipse, e é uma ferramenta que analisa classes Java e gera métricas sobre a qualidade do "Design" (projeto) para cada pacote em Java. Calcula métricas de modelo, porém diferentes das métricas da ferramenta exposta neste trabalho, pois só calcula as métricas de projetos desenvolvidos no Eclipse. O Eclipse Metric (Sourceforge 2010) é um *plug-in* para o Eclipse, que calcula um número considerável de métricas de código. Apresenta os resultados através de gráficos e tabelas, oferecendo navegabilidade, porém só calcula métricas de projetos desenvolvidos no Eclipse, ou seja, não é uma ferramenta genérica, embora seja livre. O quadro 1 resume as características das ferramentas analisadas.

ISSN 2236-8884

Quaro 1: Comparação das Ferramentas para Cálculo de Métricas

Nome	Livre	Genérica	Métricas de Modelo	Métricas de Código
SDMetrics	x	✓	✓	✓
Objecteering	x	x	✓	x
Together	x	x	✓	✓
JDepend	✓	x	✓	x
Eclipse Metric	✓	x	✓	✓
XMIMetricsTool	✓	✓	✓	x

Conclusões e Trabalhos Futuros

A contribuição deste trabalho para a sociedade como um todo é a produção de *software* mais qualitativo, menos propenso a erros, e, portanto, mais confiável de ser utilizado, uma vez que foi desenvolvida uma ferramenta visando apoiar a qualidade de *software* desde o início do seu ciclo de vida, que visa ser livre para uso comum. Dessa forma, é possível identificar precocemente prováveis erros que se propagariam para outras fases do desenvolvimento, economizando tempo, esforço e custo de correção.

Como trabalhos futuros, podem ser destacados: a extensão da capacidade de leitura da ferramenta, uma vez que a ferramenta foi testada apenas utilizando o XMI da ferramenta ArgoUml (Argo 2010). Isso requer um processo de análise e alteração no código, pois apesar de o XMI ser um padrão, há diferenças entre arquivos desse tipo gerados por distintas ferramentas CASE; a ampliação do conjunto de métricas, inclusive para outros diagramas UML; a disponibilização no portal do *software* público brasileiro; a possibilidade de informação de valores de referência para as métricas através de *baseline*; a adequação da ferramenta, para que ela atenda a todos os resultados esperados do Processo de Medição do Mps.br.

ISSN 2236-8884

Outras métricas podem ser acrescentadas à ferramenta, de acordo com as necessidades de cada empresa, e assim aumentar sua contribuição na garantia de qualidade de *software*.

A limitação da ferramenta está no fato de efetuar a leitura do XMI sem o uso de uma tecnologia específica, o que aumenta o trabalho, inclusive fazendo o código ficar mais extenso e mais difícil de ser estendido. A leitura do XMI deveria ser feita utilizando tecnologias como a EMF (Emf 2010) e a MDR (Mdr 2010), pois isso tornaria o trabalho mais dinâmico.

Referências

AMBLER, Scott W. ***Análise e Projeto Orientado a Objeto:*** Seu Guia para Desenvolver Sistemas Robustos com Tecnologia de Objetos. Tradução OswaldoZanelli. Rio de Janeiro: Infobook, 1998.

ArgoUML, Tigris. Disponível em: <http://argouml.tigris.org>. Acesso em: 29 set. 2010.

Borland. Disponível em: <http://www.borland.com/br/products/together>. Acesso em: 29 set. 2010.

CHIDAMBER, Shyam R.; KEMERER, Chris F. A Metrics Suite For Object Oriented Design. ***IEEE Transactions on Software Engineering***, v. 20, n. 6, p. 476-493, June 1994.

Emf. Disponível em: <http://www.eclipse.org/modeling/emf>. Acessado em 29 set. 2010.

GQM. Disponível em: <http://www.gqm.nl/>. Acesso em: 29 set. 2010.

JDepend, Disponível em: <http://www.eclipseplugincentral.com/Web_Links-index-req-viewlink-cid-125.html>. Acesso em: 29 set. 2010.

LORENZ, Mark; KIDD, Jeff. ***Object-Oriented Software Metrics***. Prentice Hall Object-Oriented Series, 1994.

ISSN 2236-8884

SUN Microsystems. ***MDR – Meta Data Repository Project***. Disponível em <http://mdr.netbeans.org>. Acesso em: 29 set. 2010.

MOLLER, K. H.; PAULISH, D. J. Software Metrics: A Practitioner's Guide to Improved Product Development. Los Alamitos. ***IEEE Transactions on Software Engineering***, 257p, 1993.

MPS.Br (a), Melhoria de Processo do ***Software*** Brasileiro. Disponível em: <http://www.softex.br/mpsbr/_guias/guias/MPS.BR_Guia_Geral_2009.pdf>. Acesso em: 29 set. 2010.

MPS.Br, (b), Guia de Implementação – Parte 2: Fundamentação para Implementação do Nível F do MR-MPS: 2009, maio 2009. Disponível em: <http://www.softex.br/mpsbr/_guias/guias/MPS.BR_Guia_de_Implementacao_Parte_2_2009.pdf>. Acesso em: 29 set. 2010.

OBJECTEERING. Disponível em: <http://www.objecteering.com>. Acesso em: 29 set. 2010.

ROCHA, A.R.C. Uma experiência na Definição do Processo de Desenvolvimento e Avaliação de Software segundo as normas Isso. ***Relatório Técnico ES-302;94***, COPPE-UFRJ, Rio de Janeiro, Brasil, 1994.

SDMetrics. Disponível em: <http://www.sdmetrics.com>. Acesso em: 29 set. 2010.

Sourceforge. Eclipse Metrics. Disponível em: <http://sourceforge.net/projects/eclipse-metrics>. Acesso em: 29 set. 2010.