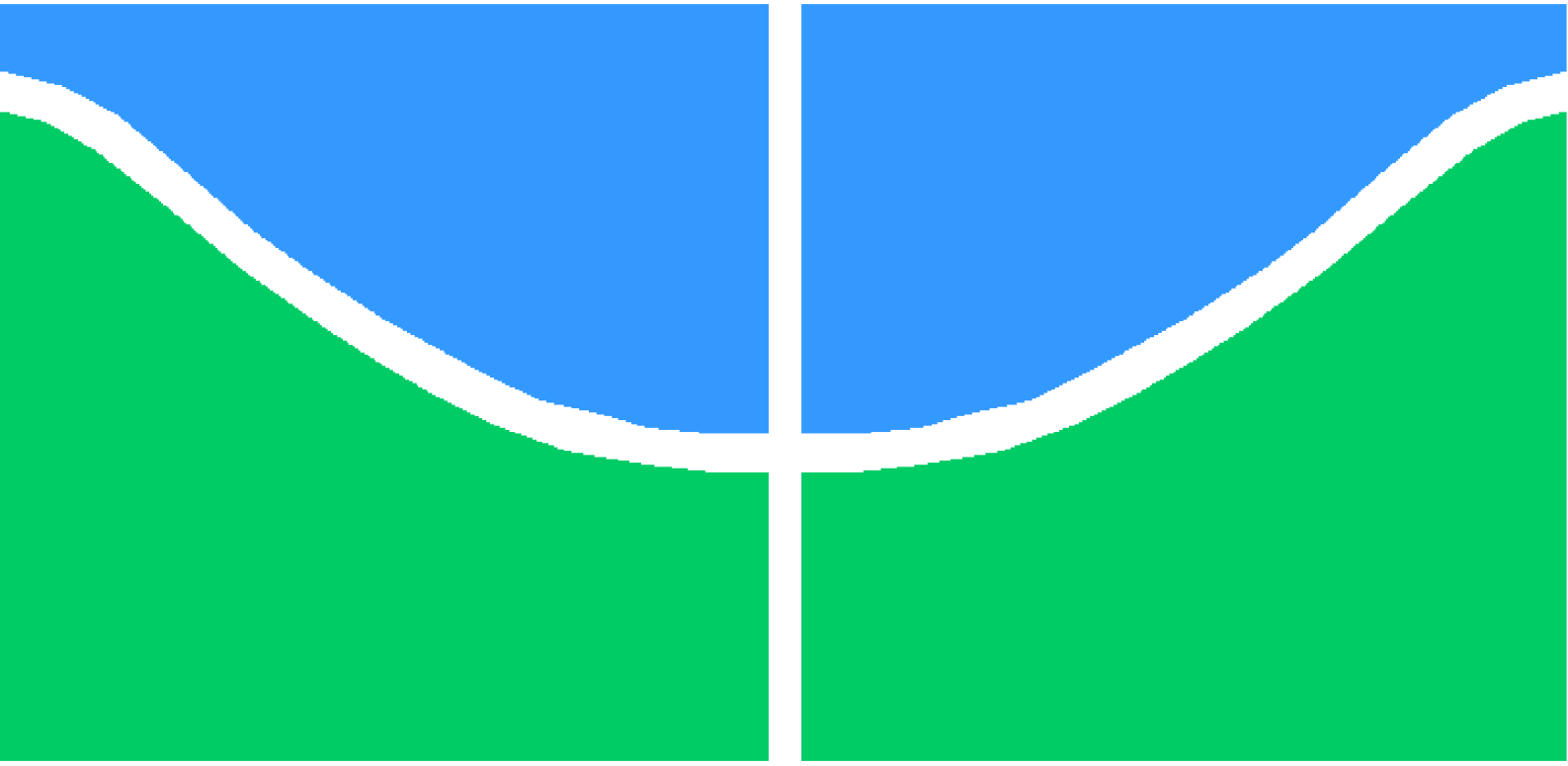

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de Software

Extração e Visualização de métricas de código-fonte em um ambiente de *Dwing*: Do Sonar ao Pentaho

Autor: Guilherme Baufaker Rêgo

Orientador: Prof. Msc. Hilmer Rodrigues Neri

Coorientador: Prof. Dr. Paulo Roberto Miranda Meirelles

Brasília, DF

2013

Guilherme Baufaker Rêgo

Extração e Visualização de métricas de código-fonte em um ambiente de *Dwing*: Do Sonar ao Pentaho

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. Msc. Hilmer Rodrigues Neri

Coorientador: Prof. Dr. Paulo Roberto Miranda Meirelles

Brasília, DF

2013

Guilherme Baufaker Rêgo

Extração e Visualização de métricas de código-fonte em um ambiente de *Dwing*: Do Sonar ao Pentaho/ Guilherme Baufaker Rêgo. – Brasília, DF, 2013-

38 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Msc. Hilmer Rodrigues Neri

Coorientador: Prof. Dr. Paulo Roberto Miranda Meirelles

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2013.

1. Palavra-chave01. 2. Palavra-chave02. I. Prof. Msc. Hilmer Rodrigues Neri. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Extração e Visualização de métricas de código-fonte em um ambiente de *Dwing*: Do Sonar ao Pentaho

CDU 02:141:005.6

Guilherme Baufaker Rêgo

Extração e Visualização de métricas de código-fonte em um ambiente de *Dwing*: Do Sonar ao Pentaho

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 01 de junho de 2013:

Prof. Msc. Hilmer Rodrigues Neri
Orientador

Prof. Dr. Paulo Roberto Miranda Meirelles
Coorientador

Titulação e Nome do Professor Convidado 01
Convidado 1

Titulação e Nome do Professor Convidado 02
Convidado 2

Brasília, DF
2013

Este trabalho é dedicado às todos que a honestidade é o maior valor que o ser humano pode vir a cultivar

Agradecimentos

Agradeço aos meus orientadores Prof. Hilmer e Prof. Paulo por todo o apoio concedido durante o Trabalho de Conclusão de Curso. Agradeço aos meus pais por terem lutado todos os dias e incansalvemente para mostrar que a educação é dos únicos bens duravéis na vida. Agradeço à companheira Luisa Helena Lemos da Cruz por todo o apoio concedido durante os dias díficeis que atendeceram a construção do presente trabalho. Agradeço a Prof. Eneida Gonzales por ter me incentivado a enfretar os desafios acreditando em minha capacidade.

'The only way of discovering the limits of the possible is to venture a little way past them into the impossible."

(Arthur C. Clarke)

Resumo

O resumo deve ressaltar o objetivo, o método, os resultados e as conclusões do documento. A ordem e a extensão destes itens dependem do tipo de resumo (informativo ou indicativo) e do tratamento que cada item recebe no documento original. O resumo deve ser precedido da referência do documento, com exceção do resumo inserido no próprio documento. (. . .) As palavras-chave devem figurar logo abaixo do resumo, antecedidas da expressão Palavras-chave:, separadas entre si por ponto e finalizadas também por ponto. O texto pode conter no mínimo 150 e no máximo 500 palavras, é aconselhável que sejam utilizadas 200 palavras. E não se separa o texto do resumo em parágrafos.

Palavras-chaves: latex. abntex. editoração de texto.

Abstract

This is the english abstract.

Key-words: latex. abntex. text editoration.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

GQM	Goal-Question-Metric
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
XP	eXtreme Programming
SCAM	IEEE International Working Conference on Source Code Analysis and Manipulation

Sumário

1 Introdução

As metodologias ágeis vem ganhando cada vez espaço no mercado global de desenvolvimento de software, pois enfatizam a qualidade do produto sobre a qualidade do processo, procurando minimizar a execução de atividades não essenciais ao longo do ciclo de vida de desenvolvimento de software.

O eXtreme Programming (XP), que é uma metodologia ágil, tem a codificação como a atividade chave durante um projeto de software (BECK, 1999). Isso também se torna perceptível por algumas práticas do XP [1] :

1. Padronização do Código: O código é a principal forma de comunicação entre a equipe, logo a padronização de código o torna consistente e fácil para todo o time ler e refatorar.

2. Propriedade Coletiva do Código: Cada programador pode melhorar qualquer parte do código quando existir a oportunidade.

3. Programação em Pares: Todo código é escrito com duas pessoas: uma que olha para uma máquina, e outra com um teclado e um mouse.

4. Integração Contínua: Todo novo código é integrado ao sistema e quando integrado, o sistema é totalmente reconstruído do zero e todos os testes devem passar ou o novo código é descartado.

Logo, em metodologias ágeis, é possível inferir a qualidade do processo de desenvolvimento, por meio do código-fonte. Um dos métodos mais utilizados é a análise estática de código, o qual segundo Emanuelsson e Nilsson (2008) significa método automático de coleta de propriedades de tempo de execução do código-fonte sem executá-lo e os resultados da análise podem servir para propósitos de verificação de erros, geração automática de casos de teste, análise de impacto ou ainda para obtenção de métricas de código-fonte.

Embora a definição formal da análise estática consolidada no âmbito acâdemico, vide os trabalhos de (WICHMANN et al., 1995) e (NIELSON; NIELSON; HANKIN, 1999), as ferramentas dísponiveis no mercado, que realizam as coletas e avaliação, ainda não suprem totalmente as necessidades de quem precisa avaliar a qualidade total do produto.

Cada ferramenta apresenta suas vantagens e limitações, contudo, de maneira geral é possível citar como limitações das ferramentas de análise estática de código:

[1] http://www.extremeprogramming.org/

1. Ausência de resultados consolidados do produto, pois não são feitas correlações entre as análises de elementos internos do código fonte (Bibliotecas, Pacotes, Classes, Métodos, Funções).

2. Ausência de mecanismos de tratamento, separação e organização de dados históricos das métricas.

3. Ausência de avaliações qualitativas sobre os valores obtidos para cada métrica, deixando assim a interpretação dos resultados restrita apenas ao nível técnico.

4. Apresentação dos dados não é visualmente agradável.

5. Grande complexidade em sua utilização.

1.1 Objetivos

Esta seção apresenta os objetivos gerais e específicos deste TCC.

1.1.1 Objetivos Gerais

Sob prisma da avaliação da qualidade de código em ambientes ágeis, neste trabalho, há como objetivo geral a prosição e construção um ambiente de *Dwing*, de modo a melhorar extração e visualização das métricas de código-fonte, suprimindo os pontos fracos enunciados acima das ferramentas de análise estática de código. O ambiente é composto de:

- SonarQube [2], que é uma ferramenta de software-livre para realizar análise estática de código

- *Data Warehouse*

- *Dashboards* com as métricas em formatos de gráficos diversos.

Este ambiente inicialmente construído para este trabalho de conclusão de curso, poderá aumentado em trabalhos futuros para a correlação de outras métricas coletadas durante o processo de desenvolvimento de software. Este ambiente também tem como objetivo a validação da abordagem desenvolvida por Meirelles (2013).

[2] http://www.sonarqube.org/

1.1.2 Específicos

Os objetivos específicos desse trabalho são:

1. Construir o ambiente de Dwing para métricas de código-fonte.
2. Validar o ambiente com o estudo de caso.
3. Incorporar visões gerenciais das métricas de código-fonte.
4. Facilitar o entendimento das métricas de código-fonte.

1.2 Organização do Trabalho

Durante a fase inicial do trabalho de conclusão de curso, a metodologia da pesquisa foi definida em pesquisa exploratória, dado que se buscava evidenciar e caracterizar o problema da visualização das métricas obtidas diretamente do código-fonte , e pesquisa biblográfica, dada a necessidade de fundamentação teórica sobre cada um dos elementos que compõem o ambiente da solução proposta. Durante a fase de construção do ambiente, utilizou-se as práticas ágeis como a realização de ciclos curtos e a integração contínua de novos pedaços da solução proposta. Nesta fase, visou-se ainda a aplicação do ambiente em um estudo de caso, de forma a a validar o ambiente proposto.

Para a primeira fase deste Trabalho de Conclusão de Curso, além desta introdução este texto está organizado em capítulos. O Capítulo 2 apresenta a fundamentação teórica sobre métricas e traz também as métricas de código-fonte, seus intervalos e definições para extração e visualização. O Capítulo 3 apresenta o SonarQube como a ferramenta escolhida para extração das métricas de código-fonte. O Capítulo 4 apresenta a fundamentação teórica do ambiente de Dwing, de modo a suprir as limitações das ferramentas, e cada um de seus componentes em detalhes. O Capítulo 5 apresenta o ambiente proposto por este trabalho para extração e visualização das métricas de código-fonte. Por fim, o Capítulo 6 apresenta o estado atual da validação do ambiente por meio de estudo de caso, bem como as atividades planejadas até o encerramento do trabalho de conclusão de curso.

2 Métricas

2.1 Processo de Medição

Medição é o mapeamento de relações empíricas em relações formais. Isto é, quantificação em símbolos com objetivo de caracterizar uma entidade por meio de seus atributos. (FENTON; PFLEEGER, 1998).

Segundo ISO/IEC 15939 (2002), medição, que é uma ferramenta primordial para gerenciar as atividades do desenvolvimento de software e para avaliar a qualidade dos produtos e a capacidade de processos organizacionais, é um conjunto de operações que visam por meio de um objeto determinar um valor a uma medida ou métrica.[1].

O processo de medição da ISO/IEC 15939 (2002), como é mostrado na figura 1, é um processo que consiste de quatro atividades que são sequenciadas em um ciclo iterativo, permitindo feedback e melhoria contínua do processo. Ele é uma adaptação do ciclo PDCA (Plan-Do-Check-Act), comumente utilizado como base para a melhoria da qualidade (BARCELLOS, 2010). O processo visa a construção de produtos de informação para cada necessidade seguindo um modelo de informação, que é mostrado na figura 2.

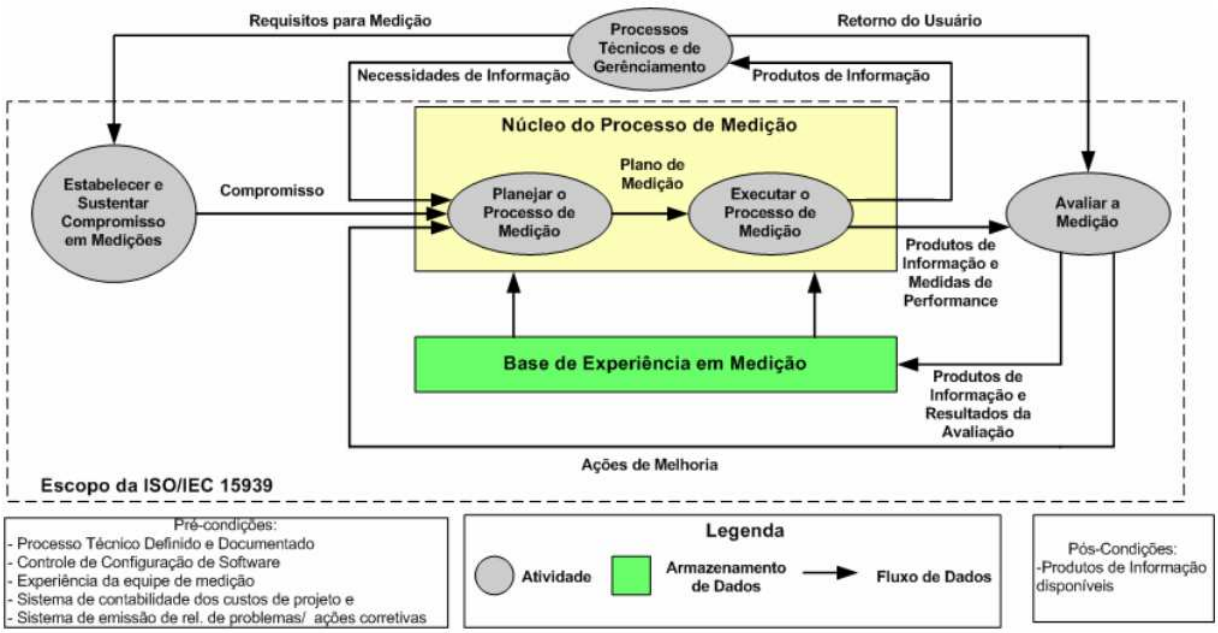

Figura 1 – Processo de Medição da ISO 15939 - Extraído de (GAVA et al., 2006)

[1] A definição formal da ISO/IEC 15939 (2002) não utiliza o termo métrica, sendo que este é definido pelo GQM em Basili, Caldiera e Rombach (1996), que é uma outra abordagem para medição, contudo compreende-se que o termo medida tem valor semântico equivalente ao de métrica no contexto do presente trabalho

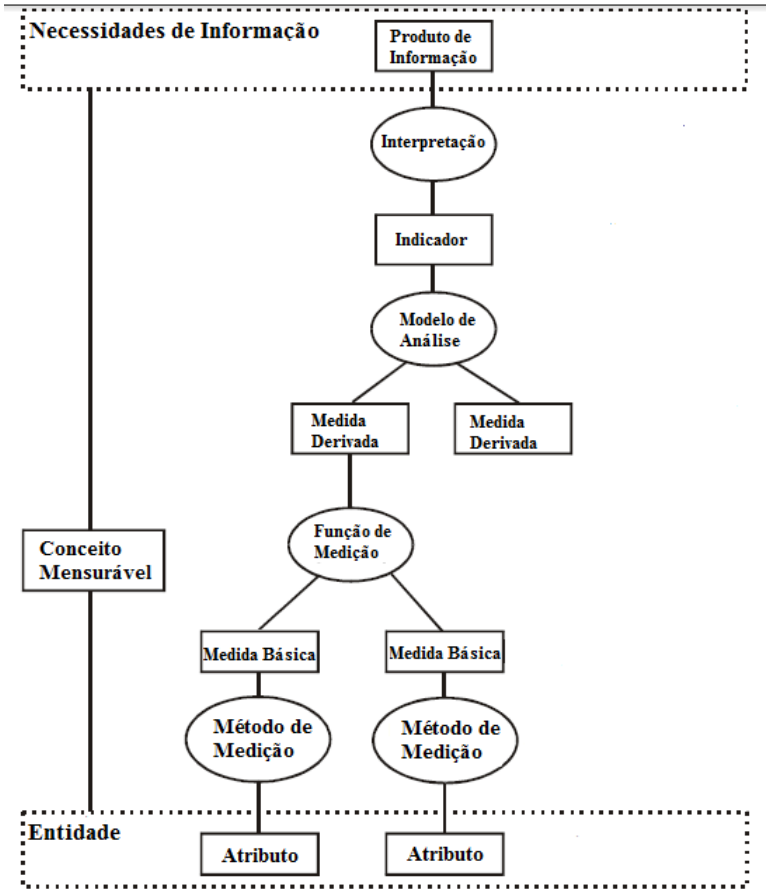

Figura 2 – Modelo de Informação da ISO 15939

A terminologia do modelo de informação, que é mostrado na figura 2 é descrito a seguir:

Necessidade de Informação. São situações as quais requer conhecimento de uma ou mais entidades para gerenciar objetivos, metas, riscos e problemas.

Entidade. Uma entidade é um objeto (como por exemplo, um processo, produto, projeto ou recurso) que é caracterizado por um atributo mensurável;

Atributo. Um atributo é uma propriedade ou característica de uma entidade que pode distiguí-la quantitativamente ou qualitativamente utilzando métodos manuais ou automáticos

Conceito Mensurável. Uma relação entre os atributos e as necessidades de informação. Como por exemplo, a qualidade de um produto.

Medida Básica ou Métrica Direta. Uma medida básica é definida em termos do atributo e do método para sua quantificação, ou seja, é a variável a qual se atribui um valor. Cada medida básica é funcionalmente independente de outras medidas básicas e captura apenas um atributo.

Método de Medição. O método de medição é uma sequência de operações, descritas genericamente, usadas qualificar ou quantificar um atributo, ou seja, obter uma medida básica ou derivada com respeito a uma escala. Um método de medição pode ser classificado em:

Subjetivo. Envolve o julgamento humano. Resulta em medidas ou métricas subjetivas.

Objetivo. Baseado em regras númericas, tais como contagem. Pode ser realizado de forma manual ou automática. Resulta em medidas ou métricas objetivas.

Escala. Uma escala é um conjunto ordenado de valores, contínos ou discretos, ou uma série de categorias as quais um atributo é mapeado. O método de medição mapeia a magnitude, ou valor absoluto, de um atributo medido em um valor de escala. As escalas podem ser:

Nominal. A medição é categórica. Por exemplo, a classificação dos tipos de defeitos. Nesta escala, só é possível realização de comparações.

Ordinal. A medição é baseada em ordenação. Por exemplo, a classificação das *releases*. Nesta escala, é possível realizar ordenação de elementos.

Intervalo. A medição é baseada em distâncias iguais definidas para as menores unidade. Por exemplo, o aumentar de 1º C em um dia corresponde ao mesmo 1º C em outro dia qualquer. Nesta escala é possível realizar ordenação, soma e subtração.

Racional. A medição é baseada em distâncias iguais definidas para as menores unidades, e neste caso é possível a ausência por meio do zero absoluto. Como por exemplo, a quantidade de linhas de código em uma classe. Nesta escala, é possível realizar ordenação, soma, subtração, mutiplicação e divisão.

Função de Medição. Trata-se do algoritmo para combinar duas ou mais medidas básicas ou métricas diretas.

Medida Derivada ou Métrica Derivada. Uma medida derivada ou métrica derivada é definida como uma função de medição de duas ou mais medidas básicas ou métricas diretas.

Indicador. É uma medida que fornece uma estimativa ou avaliação de atributos especificados em relação a uma necessidade de informação. Um indicador inclui um ou mais valores de medidas básicas e/ou derivadas.

Interpretação. É a explicação dos indicadores, isto é, a interpretação que cada valor quantitativo do indicador mostra.

Produto de Informação É o resultado esperado do processo de medição, que visa responder as necessidades de informação. O produto de informação pode vir a se materializar em forma de relatórios, gráficos ou planilhas.

2.2 Classificação das Métricas

O modelo proposto pela ISO/IEC 15939 (2002) concentra-se na identificação da necessidade de informação sobre alguma entidade (processo ou projeto). Daí advém a forma mais genérica de classificação, quanto a objeto da métrica, que divide as métricas de software em: *métricas de processo* e *métricas de produto* (MILLS, 1999). Ainda é possível, segundo a ISO/IEC 15939 (2002), dividir as métricas quanto ao método de medição, podendo estas serem *métricas objetivas* ou *métricas subjetivas*.

As métricas de produto, segundo o modelo de qualidade da ISO/IEC 25023 (2011) (Fig. 3), podem ser subdivididas em três categorias.

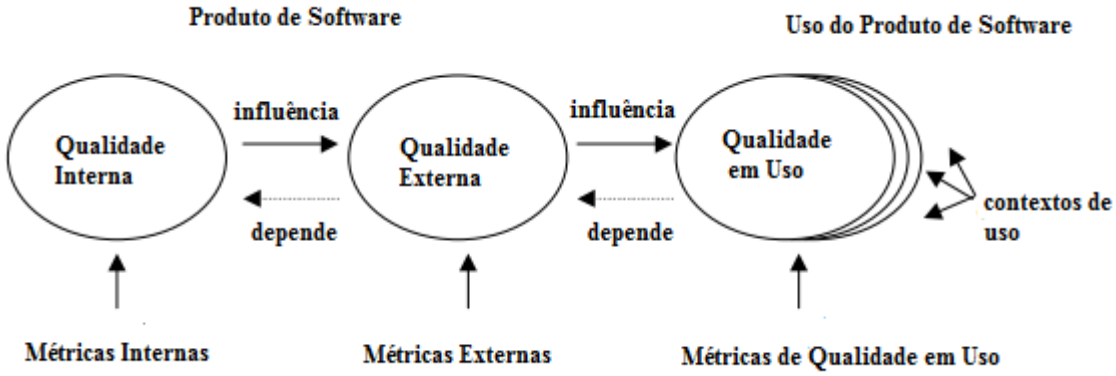

Figura 3 – Modelo de Qualidade do Produto da ISO 25023

Métricas Internas. São métricas que aferem a qualidade interna do software que é avaliação de estruturas internas que compõem o software em estágio de desenvolvimento. São exemplos de atributos da qualidade interna: simplicidade, concisão, coesão, clareza, baixo acoplamento e generalidade.

Métricas Externas. São métricas que capturam o comportamento do software. Exemplos de atributos da qualidade externa: correção, usabilidade, eficiência e robustez. qualidade externa mede o comportamento do software. Estas só podem ser aferidas por atividades de teste do desenvolvimento do software em condições similares as que serão encontradas em ambientes de implantação.

Métricas de Qualidade em Uso. São métricas que aferem se o software atende as necessidades do cliente com eficiência, produtividade, segurança e satisfação em con-

textos específicos de uso. Estas só podem ser coletadas em ambientes reais, isto é, o ambiente o qual fora previamento projetado.

2.3 Métricas de Código-Fonte

Mills (1999) define que o produto de software deve ser entendido como um objeto abstrato que evolui de um conjunto inicial de necessidades até o sistema software acabado, incluindo o código-fonte e várias outras formas de documentação.

Segundo a chamada de trabalhos do SCAM 2013 [2], código-fonte é "qualquer descrição completamente executável de um sistema de software, desde de linguagem de máquina até representações gráficas executáveis".

Segundo Sommerville (2010), há duas formas de executar verificação e validação no código-fonte:

Estatica. A forma estática de avaliação do código-fonte é o meio pelo qual se pode analisar as estruturas internas do código sem realizar sua execução (WICHMANN et al., 1995) e (NIELSON; NIELSON; HANKIN, 1999). Popularmente, o termo análise estática de código se refere uma análise automatizada, que é uma das técnicas estáticas de inspeção de software para obtenção de métricas de código-fonte (TERRA; BIGONHA, 2008) (EMANUELSSON; NILSSON, 2008).

Dinâmica. A forma dinâmica de avaliação do código-fonte requer a execução da implementação. Os testes de software são técnicas que compreendem a abordagem de verificação dinâmica, podendo ser estes o teste de defeitos que tem a finalidade de encontrar inconsistências entre um programa e sua especicação, isto é, revelar defeitos e o teste de validação que tem a finalidade de mostrar que o software tem o comportamento que o cliente deseja (SOMMERVILLE, 2010).

Tendo como base a seção 2.2, métricas de código-fonte podem ser classificadas em *métricas internas de produto*, quando obtidas por meio da análise estática do código-fonte, *métricas externas de produto*, quando obtidas por resultados de testes de software. As duas categorias podem ser classificadas em *métricas objetivas*, pois o método de medição, tanto para métricas internas quanto externas, é definido por meio de regras matemáticas que são, em sua grande maioria, automatizadas em sua forma de coleta.

2.3.1 Métricas Internas de Código-Fonte

Esta subseção traz a classificação de métricas internas de código-fonte. O mesmo pode ser caracterizado em subcategorias:

[2] 13th IEEE International Working Conference on Source Code Analysis and Manipulation

Tamanho. O tamanho do código-fonte foi uma das primeiras categorias a ser concebida, dado que o software poderia ocupar espaço tanto em forma de cartões perfurados quanto em forma de papel quando o código-fonte é impresso. A seção 2.3.1.1 apresenta as principais métricas de tamanho.

Complexidade. A primeira lei de Lehman (1980) diz que um software deve ser continuamente adptado, se não a satisfação cai progressivamente. Contudo a segunda lei de Lehman (1980) enuncia que a complexidade aumenta à medida que o software é evoluído, a menos que seja um trabalho de manutenção. Logo é perceptível que as métricas de complexidadde estão diretamente ligadas as métricas de tamanho, sendo a modificação em uma provavelmente impactará na outra. A seção 2.3.1.2 apresenta as principais métricas de complexidade.

Paradigma de Programação. A evolução dos paradigmas de programação permitiu que as linguagens de programação, assumissem diversas características entre si. O paradgima funcional, por exemplo, enxerga o programa como uma sequência de funções que podem ser executadas em modo funcional. Já o paradigma orientado a objetos visa abstrair as unidades computacionais em Classes, que representam em grande parte do desenvolvimento unidades reais, e partir destas abstrair instâncias computacionais, isto são, objetos propriamente ditos. Para cada paradigma, são necessárias métricas específicas. A seção 2.3.1.3 apresenta as principais métricas do paradigma orientado a objetos, pois este ainda é o paradigma mais utilizado no mercado de desenvolvimento de software.

2.3.1.1 Métricas de Tamanho

A primeira métrica utilizada para medir o tamanho do software é **Linhas de código** (LOC - Lines of Code) que indica o número de linhas não brancas ou não comentadas de código-fonte.

2.3.1.2 Métricas de Complexidade

2.3.1.3 Métricas Orientadas à Objetos

3 SonarQube

3.1 Descrição da Ferramenta

O SonarQube [1], ateriormente conhecido como Sonar, é uma ferramenta que realiza coleta de dados do código-fonte. Em sua documentação, são citados sete eixos de controle de qualidade cobertos na ferramenta:

1. Duplicação de Código
2. Detecção de Complexidade de Código
3. Parão de Codificação
4. Testes Unitários
5. Arquitetura e Projeto
6. Potenciais Defeitos
7. Comentários

3.1.1 Escolha do SonarQube

O SonarQube foi escolhido como ferramenta para extração de dados do código-fonte devido as seguintes características:

Ferramenta Livre: O SonarQube é uma ferramenta livre que está licenciada sob a GNU LPGL 3, que é uma das licenças de software livre.

Suporte a várias Linguagens de Programação: É realizada coleta em mais de 20 linguagens de programação, como por exemplo, Java, C#, Python, C++, .Net, PHP e entre outras.

Mutiplataforma: Há suporte para os sistemas operacionais Windows, Linux, Mac OS e há também para sevidores de aplicação Java, como por exemplo, o Apache Tomcat.

Integração com Ferramentas de Integração Contínua: é possível realizar a integração com ferramentas de integração contínua, como por exemplo, Jenkins. Assim a cada versão bem sucedida de construção do código-fonte, é possível aferir a qualidade do código.

[1] Toda a documentação e *downloads* estão disponíveis em http://www.sonarqube.org/

Persistência em Base de Dados: toda a coleta de dados é persistida em uma base de dados. Há suporte para as principais ferramentas de bancos de dados utilizadas no mercado: MySQL, PostgreSQL e Oracle.

Interface Amigável: Diferentemente de outras ferramentas disponíveis no mercado, o SonarQube tem inferface web, ou seja, é possível visualização das métricas do código-fonte seja realizada em navegador a escolha do usuário. É possível ainda visualizar todo o código-fonte analisado.

Integração com Ferramentas de Gestão de Defeitos: É possivel integrar com ferramentas de gestão de defeitos, tais como Mantis e JIRA.

Suporte a Mutiprodutos: Há suporte para visualização de métricas de mais um Produto.

Suporte a Multi-Idiomas: Há suporte para Inglês, Japonês, Russo, Francês, Italiano, Espanhol e Português.

Diversidade de Plugins: O SonarQube permite a extensão de funcionalidades por meio de plugins, sendo que há uma variedade de plugins para mais diversas necessidades disponíveis para *download*. Cabe ressaltar que alguns são gratuitos e outros tem licença comercial.

Presença de Web Service: Há um Web Service que extrai as métricas disponíveis para diversos formatos, tais como JSON, XML e CSV.

3.1.2 Limitações Observadas

Após uma análise ostensiva na ferramenta, concluiu-se que o SonarQube apresenta as seguintes limitações:

Ausência de Visão Consolidada de Métricas: A maior parte das métricas que SonarQube estão apenas no âmbito de Classes ou Pacotes.

Impossibilidade Definição de Mútiplos Intervalos: O sonarQube não permite a configuração de múltiplos intervalos. É definido apenas, um intervalo de alerta (amarelo) e um intervalo de atenção (vermelho).

Ausência de Avaliações Qualitativas: As métricas não trazem nenhuma avaliação qualitativa associada, ou seja, sem o conhecimento específico do significado de cada métrica, a interpretação dos valores obtidos não é representativo nem elucidativo sobre a qualidade do código.

Métricas como Violações: O SonarQube utiliza para coletar dados de código fonte, em algumas linguagens de programação, coletores que são identificadores padrão de codificação. Isso quer dizer que algumas métricas não são aprsentadas com valor absoluto, apenas são indicadas com violações à regras estabelecidas pelos padões de codificação.

Ausência de Representações Estátisticas das Métricas: O SonarQube não apresenta nenhum suporte a análise estátistica de dados.

Referências

BARCELLOS, M. P. Medição de software - um importante pilar da melhoria de processos de software. *Engenharia de Software Magazine*, 2010. Citado na página 27.

BASILI, V. R.; CALDIERA, G.; ROMBACH, H. D. *The Goal Question Metric Approach*. [S.l.]: Encyclopedia of Software Engineering, 1996. Citado na página 27.

BECK, K. *Extreme Programming Explained*. [S.l.]: Addison Wesley, 1999. Citado na página 23.

EMANUELSSON, P.; NILSSON, U. A comparative study of industrial static analysis tools. *Electron. Notes Theor. Comput. Sci.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 217, p. 5–21, jul. 2008. ISSN 1571-0661. Citado 2 vezes nas páginas 23 e 31.

FENTON, N. E.; PFLEEGER, S. L. *Software Metrics: A Rigorous and Practical Approach*. 2 edition. ed. [S.l.]: Course Technology, 1998. 656 p. Citado na página 27.

GAVA, V. L. et al. Modelo do processo de medição de software: pré-condições para sua aplicação. In: *XXVI Encontro Nacional de Engenharia de Produção*. [S.l.: s.n.], 2006. Citado 2 vezes nas páginas 15 e 27.

ISO/IEC 15939. *ISO/IEC 15939: Software Engineering - Software Measurement Process*. [S.l.], 2002. Citado 2 vezes nas páginas 27 e 30.

ISO/IEC 25023. *ISO/IEC 25023: Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Measurement of system and software product quality*. [S.l.], 2011. Citado na página 30.

LEHMAN, M. M. Programs, life cycles, and laws of software evolution. *Proc. IEEE*, v. 68, n. 9, p. 1060–1076, September 1980. Citado na página 32.

MEIRELLES, P. R. M. *Monitoramento de métricas de código-fonte em projetos de software livre*. Tese (Doutorado) — Instituto de Matemática e Estátistica – Universidade de São Paulo (IME/USP), 2013. Citado na página 24.

MILLS, E. E. Metrics in the software engineering curriculum. *Ann. Softw. Eng.*, J. C. Baltzer AG, Science Publishers, Red Bank, NJ, USA, v. 6, n. 1-4, p. 181–200, abr. 1999. ISSN 1022-7091. Citado 2 vezes nas páginas 30 e 31.

NIELSON, F.; NIELSON, H. R.; HANKIN, C. *Principles of Program Analysis*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1999. ISBN 3540654100. Citado 2 vezes nas páginas 23 e 31.

SOMMERVILLE, I. *Software Engineering*. 9. ed. Harlow, England: Addison-Wesley, 2010. ISBN 978-0-13-703515-1. Citado na página 31.

TERRA, R.; BIGONHA, R. S. Ferramentas para análise estática de códigos java. Departamento de Ciência da Computação - UFMG, 2008. Citado na página 31.

WICHMANN, B. et al. Industrial perspective on static analysis. *Software Engineering Journal*, 1995. Citado 2 vezes nas páginas 23 e 31.