FERRAMENTA DE AUXILIO À MANUTENÇÃO DE SOFTWARE ATRAVÉS DA COLETA DE MÉTRICAS DE REPOSITÓRIOS DE CÓDIGO FONTE

Software Maintenance Support Tool Through Collecting Metrics from Source Code Repository

Anna Maria Greco Carvalho[1], Angelo Cesar Mendes da Silva[2], José Augusto Rodrigues Lima[3] e Marco Antônio Pereira Araújo[4]

Resumo: Procurando controlar a evolução de um software com o intuito de prolongar sua vida útil e melhorar sua qualidade, este artigo propõe uma ferramenta que ao coletar métricas de suas versões permita aos mantenedores do software uma visualização e análise da evolução sofrida e forneça indícios do seu futuro. A ferramenta foi desenvolvida utilizando a linguagem de programação Java, as bibliotecas SVNKit para o download das versões do software e o CyVis para a coleta e cálculo das métricas de complexidade ciclomática e número de linhas de código fonte. A ferramenta realiza o download das versões do software, calcula as métricas e possibilita visualizar sua evolução no decorrer das manutenções realizadas, permitindo um comparativo com o número de classes e métodos desenvolvidos. O uso da ferramenta simplifica a análise da evolução do software e auxilia na tomada de decisões quanto a forma de manutenção a ser empregada, influindo para uma maior qualidade de software.

Palavras-chave: evolução de software, manutenção de software, manutenção evolutiva.

Abstract. *With the intension to control a software evolution, in order to prolong its useful life and increase its quality, this paper proposes a tool that by collecting metrics from its versions, allows the software maintainers to view and analyze the evolution occurred as well as presents some evidences of its future. The tool was developed by using the programming language Java, the SVNKit library for the software versions download and the CyVis to collect and calculate the cyclomatic complexity and number of lines of source code metrics. The tool downloads the software versions, calculates the metrics and displays the software*

[1] Curso Técnico em Informática, FAPEMIG/IFSUDESTEMG, annagreco1997@hotmail.com
[2] Bacharelado em Sistemas de Informação, Bolsista CAPES, programa Jovens Talentos, angelo_cms@yahoo.com.br
[3] Bacharelado em Sistemas de Informação, FAPEMIG/IFSUDESTEMG, josedelimajf@gmail.com
[4] Núcleo de Informática, marco.araujo@ifsudestemg.edu.br

evolution during the maintenance, allowing a comparison between the class numbers and the developed methods. The use of the tool simplifies the analysis of the software evolution, helps at taking decision about the way of maintenance is to be used, inducing to a higher software quality.

Keywords: *Software evolution, software maintenance, evolutive maintenance.*

INTRODUÇÃO

Quando um software está inserido num ambiente do mundo real, precisa se adaptar continuamente às novas necessidades e requisitos desse ambiente para não se tornar obsoleto. Com isso, é preciso que seja sempre submetido à manutenção. Na busca por qualidade de software, surge a necessidade de controlar e avaliar o desenvolvimento do sistema de forma que prolongue seu tempo de vida útil, assim como influenciar positivamente em sua eficiência.

Mantendo o controle sobre a manutenção, desempenho e a capacidade de evolução de um sistema, possibilita-se que novas funções demandadas pelo ambiente em que ele está inserido serem implementadas mais facilmente. Toda essa análise é útil para manutenções adaptativas na qual se possui a necessidade de modificar o software para um novo ambiente de trabalho, bem como também para manutenções evolutivas, onde se adiciona ao software novas funções não projetadas em seu escopo inicial, conforme os tipos de manutenção apresentados por PRESMANN (2011).

Este projeto propõe o desenvolvimento de uma ferramenta coletora de métricas das versões de um software que tenha sofrido manutenções ao longo do tempo, permita a seus mantenedores uma visualização e análise da sua evolução. A ferramenta fornece indícios do futuro do software ao revelar dados que permitem ao mantenedor tomar decisões que resultem na maior qualidade do software. Para auxiliar nesse objetivo, esse estudo foi baseado nas Leis de Evolução de Software (LSE) propostas por LEHMAN et al. (1997) e na coleta de métricas de software, com intuito de avaliar a aplicação das Leis aos sistemas atuais, analisando perspectivas de melhoria no software e a possibilidade de tomada de decisão baseados em dados quantitativos, através da coleta dos dados ao longo de sua evolução.

Os dados analisados são resultantes da aplicação de técnicas para coleta de métricas de software em um sistema. Métrica significa uma indicação quantitativa de um ou mais itens específicos através de uma unidade de medida padronizada. Medição é o ato ou processo de medir, ou seja, de se obter a métrica (PRESSMAN, 2011).

Com o intuito de apoiar evolução de software, este trabalho apresenta uma ferramenta que foi desenvolvida para permitir um estudo mais aprofundado através da análise detalhada sobre o processo de evolução de software. Essa ferramenta é baseada nos conceitos e técnicas do cálculo de algumas métricas em Engenharia de Software, e seu desenvolvimento ocorreu utilizando a linguagem Java e ferramentas *open source*.

Este trabalho está dividido em mais quatro seções além desta introdução. A seção 2 aborda os materiais e os métodos utilizados para o desenvolvimento. Na seção 3, é apresentada a ferramenta desenvolvida pela pesquisa junto aos resultados das métricas coletadas. A seção 4 apresenta o uso da ferramenta desenvolvida em um estudo experimental, com os resultados dos dados coletados pela ferramenta no estudo, e a análise dos mesmos. Por fim, na seção 5 são apresentados os resultados obtidos até o presente momento e os trabalhos futuros a serem desenvolvidos.

MATERIAL E MÉTODOS

Para compreender o estudo da evolução de software, é importante considerar os estudos de LEHMAN et al. (1997) e a criação das Leis de Evolução de Software. É intenção do trabalho poder comparar os resultados obtidos com a ferramenta, com a teoria que envolve a evolução de sistemas de software. São oito leis no total, aplicáveis a qualquer software que implemente uma solução computacional no mundo real:

- Mudança Contínua: sugere que um software sofre envelhecimento. Sistemas são feitos para refletir o mundo real e o mundo está em constantes mudanças, logo se um sistema não se adapta às novas exigências do ambiente, perderá sua qualidade e se tornará cada vez menos satisfatório;
- Incremento da Complexidade: uma vez implementadas modificações em um sistema, esse tende a ter sua estrutura fragmentada, há crescimento das interações entre os elementos do sistema e sua complexidade aumenta até ser necessário um trabalho de simplificação e reestruturação que demanda recursos extras;
- Auto regulação: a evolução de um software é garantida pelo trabalho de uma equipe de técnicos que trabalham de forma a seguir prazos e pontos de controle estipulados pela organização, tais mecanismos de estabilização, ao longo do tempo, definem uma dinâmica que torna constante o esforço incremental gasto em cada versão, dessa forma, atributos do sistema como tamanho, tempo entre versões e números de defeitos relatados são aproximadamente invariáveis para cada sistema ao longo do tempo;

- Estabilidade Organizacional: a longo prazo, essa taxa torna-se constante e independe dos recursos dedicados ao desenvolvimento do sistema;
- Conservação de Familiaridade: a taxa e qualidade de progresso são influenciadas pela taxa de aquisição de informações necessárias pela equipe responsável pela evolução do sistema. A familiaridade de todos os integrantes da equipe com os objetivos é um fator determinante na evolução do software. Sendo assim, quanto mais mudanças, maior a dificuldade de que toda equipe esteja ciente dos objetivos e quanto mais incremento e acréscimo de novas funcionalidades maior o número de defeitos;
- Crescimento Contínuo: após implementado um sistema, mudanças serão necessárias para a satisfação do cliente. Tais mudanças não foram consideradas antes pela equipe desenvolvedora e assim são necessárias aplicações externas e módulos extras gerando um aumento do conteúdo funcional do sistema;
- Declínio da Qualidade: se um sistema não se adapta às modificações e exigências do ambiente, com o passar do tempo, seu nível de satisfação decresce, sofre um envelhecimento e sua qualidade consequentemente decai;
- Sistemas de *Feedback*: um software é constantemente realimentado por retorno positivo ou negativo de suas características. O crescimento do sistema é estabelecido pela quantidade de retornos positivos e negativos, assim sistemas de retorno são utilizados para aprimoramentos significativos do produto.

Métrica de software é um termo utilizado para descrever medidas quantitativas que permitem avaliar a qualidade e produtividade de um software de acordo com PRESSMAN (2011). Para este trabalho, foram utilizadas três métricas: a complexidade ciclomática, desenvolvida por McCABE (1976), que extrai dados quantitativos do total de instruções de execução, utilizada para indicar complexidade estrutural de um programa, o número de instruções (LOC - *Lines of Code*) que consiste na contagem da quantidade do número de código de um software, revelando o comprimento do código e o tamanho do programa e a quantidade de métodos que o software possui (NOM – *Number of Methods*). Tais métricas são importantes na avaliação de cada uma das Leis de Evolução de Software.

Existem diversas ferramentas para análise e coleta de métricas em software que auxiliam na manutenção e estabelecem um produto de boa qualidade. O Projeto JaBUTi (*Java Bytecode Understanding and Testing*) (2015) é uma ferramenta de suporte ao teste estrutural para programas Java, desenvolvida pelo grupo de Engenharia de Software da USP-São

Carlos. Nela estão implementados critérios de testes de software baseados em fluxo de controle e critérios baseados em fluxo de dados. O *Metrics* (2015) é um *plugin* para o ambiente *Eclipse* que mede várias métricas com média e desvio padrão, e detecta ciclos em dependências de pacotes, as métricas coletadas podem ser usadas para elaborar gráficos. A ferramenta *Analyst4j* (2015) é baseada no *Eclipse*, disponível como uma ferramenta *stand-alone* ou *plugin* para o *Eclipse*. Coleta métricas, analisa a qualidade e gera um relatório sobre software Java. O programa CKJM (*Chidamber and Kemerer Java Metrics*) (2015) calcula as métricas processando o *bytecode* de arquivos Java compilados.

A Tabela 1 apresenta as métricas que constam nas ferramentas citadas e também são importantes no contexto deste trabalho.

Software/Métrica	CC	LOC	NOM
JaBUTI	x	x	x
Metrics	x	x	
CKJM		x	x
Analyst4j	x	x	x

Tabela 1: Comparativo de métricas

Entretanto, as ferramentas identificadas não possuem o recurso de integração com repositórios *online* de códigos fonte para *download* de alterações, fornecendo análise apenas durante o processo de compilação, sendo esse um requisito fundamental para a proposta deste trabalho, justificando o desenvolvimento da ferramenta proposta.

A FERRAMENTA

A ferramenta proposta neste trabalho consiste em uma aplicação que realiza automaticamente o *download* de todas as versões de um software contidas em um repositório de código fonte. A ferramenta foi desenvolvida em linguagem de programação Java, foram também utilizadas as bibliotecas SVNKit para o download das versões do software a ser analisado e ainda foi utilizado para a coleta e cálculo das métricas, a CyVis (2014), uma ferramenta *open source* feita em Java que extrai e analisa métricas de arquivos *class* do software colhido no repositório, retornando as métricas de complexidade ciclomática, número de instruções e o número de métodos de uma classe, a nível de projeto, pacote e classe.

A ferramenta é utilizada para analisar a evolução de um software que sofreu manutenção ao longo do tempo, ao realizar o *download* das versões do software através de um

repositório SVN, um sistema de controle de versão *open source* criado pela *CollabNet* e atualmente desenvolvido como um projeto da *Apache Software Foundation*. Calcula as métricas de cada versão necessitando que o usuário forneça alguns dados do projeto, como nome do projeto, URL do repositório onde se encontra o código fonte, *login* e a senha do repositório, além da pasta de destino onde as versões serão salvas. A Figura 1 mostra a tela inicial da ferramenta.

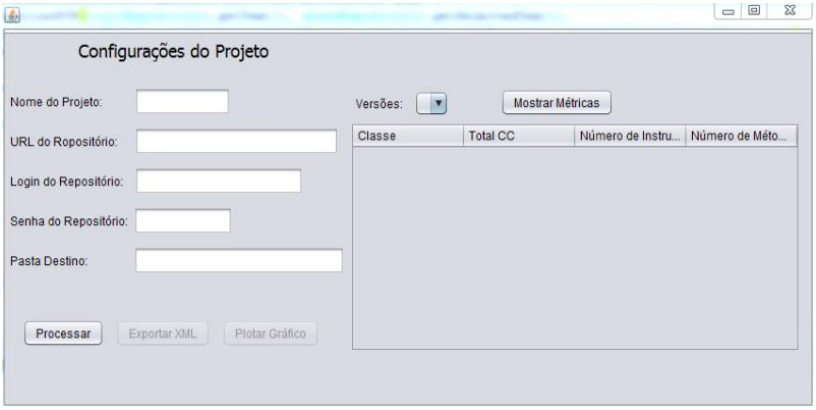

Figura 1 – Tela Inicial da Ferramenta.

Após o *download*, é realizada a extração das métricas de complexidade ciclomática e linhas de código fonte, tendo as opções de exportação dos dados das métricas para um arquivo XML, ou a representação gráfica da evolução dessas métricas do software de acordo com as implementações das novas versões. Com isso, possibilita visualizar sua evolução no decorrer das manutenções realizadas, como mostra a Figura 2, permitindo um comparativo com o número de classes e métodos desenvolvidos. Permite ainda que o usuário verifique as métricas de cada classe separada em sua respectiva versão.

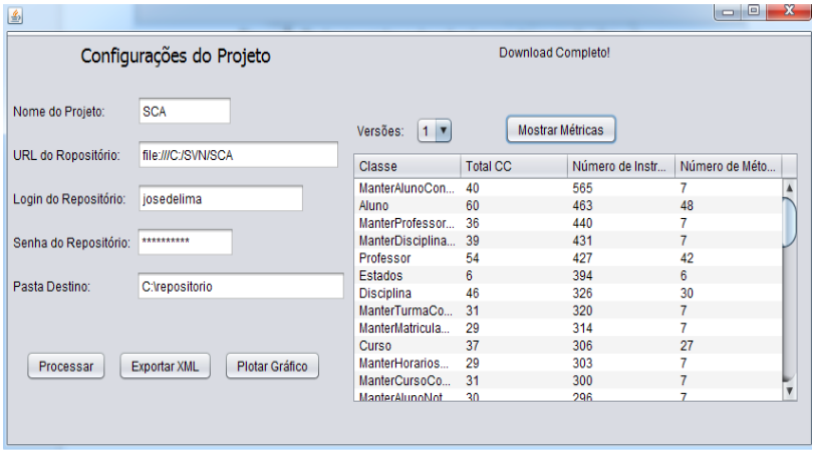

Figura 2 – Ferramenta após o *download* das métricas.

A ferramenta também gera um gráfico com dados das métricas (Figura 3), obtido através dos botão “Plotar Gráfico”, revelando o comportamento do software ao longo de suas versões. Através do botão “Exportar XML” é criado um arquivo XML onde são exportados os dados quantitativos das métricas.

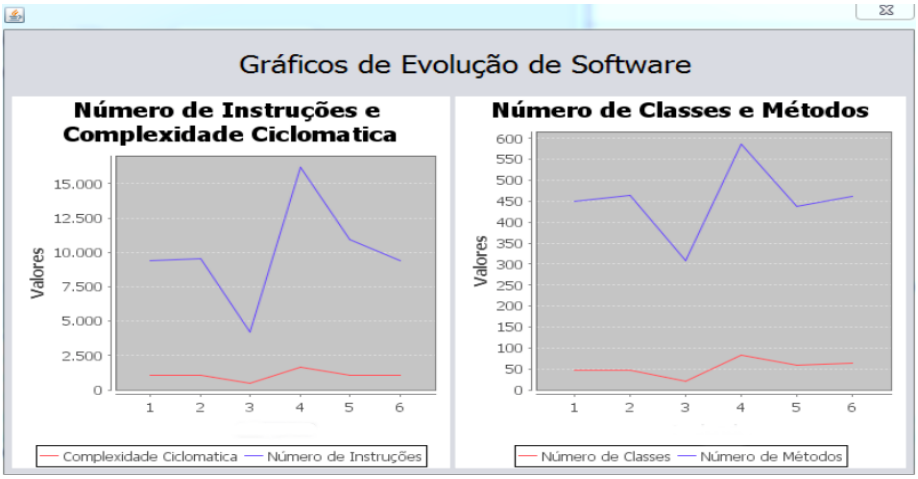

Figura 3 – Gráfico gerado pela ferramenta.

RESULTADOS E DISCUSSÃO

Com a aplicação da ferramenta em um sistema que sofreu evolução ao longo do tempo, foi possível analisar, através dos gráficos gerados, que o sistema se comportou de acordo com algumas Leis de Evolução de Software, propostas por LEHMAN et al. (1997), o que permitiu comparativos com estudos da literatura técnica. Foi observada a Lei da Mudança Contínua, que dita que um software deve estar em constante mudança para não se tornar obsoleto, visto que o software sofreu relevantes alterações ao longo do tempo, nos valores de classes, métodos, linhas de instrução e complexidade ciclomática, o que indica que ele está sendo modificado. Segundo LEHMAN et al. (1997), um sistema que não se adapta a novas exigências do ambiente, perderá sua qualidade e se tornará cada vez menos útil.

Outra Lei de Evolução de Software que pode ser analisada é a influência da Lei de Incremento da Complexidade, onde se afirma que uma vez implementadas modificações em um sistema, esse tende a ter sua estrutura fragmentada, havendo crescimento das interações entre os elementos do sistema e sua complexidade aumenta até ser necessário um trabalho de simplificação e reestruturação que demanda recursos extras, a chamada manutenção preventiva, ou até mesmo um processo de reengenharia. Embora não muito enfática, pode-se perceber um aumento da complexidade do sistema através do aumento da medida da complexidade ciclomática, o que pode vir a confirmar essa Lei.

CONCLUSÕES

O uso da ferramenta simplifica a análise e o estudo da evolução do software e auxilia na tomada de decisões quanto à forma de manutenção a ser empregada, influindo para uma maior qualidade de software e demandando menos custo. Também traz uma automação do *download* de um sistema e do cálculo de algumas de suas métricas, podendo traçar um gráfico das mudanças que ocorreram em um projeto de um software.

O projeto, no entanto, ainda se encontra em desenvolvimento e tem como objetivo ainda a implementação de novas funções como análise estatística dos dados obtidos de cada manutenção, gráficos de evolução a partir dos dados coletados levando em consideração as Leis de Evolução de Software e a implementação de funcionalidade para verificar sua validade.

Agradecimentos

Os autores agradecem à FAPEMIG e ao IF Sudeste MG pelo apoio financeiro e incentivo à pesquisa intitulada “Desafios em Manutenção de Software Evolutiva: avaliação, impactos e oportunidades de pesquisa”, na qual este trabalho está inserido.

BIBLIOGRAFIA

Analyst4j. Disponível em: http://www.ijser.org/paper/Analysis-of-software-Metrics-Tools-A Survey-Approach.html. Acessado em: 22/10/2015.

Apache Subversion. Disponível em: http://subversion.apache.org/. Acessado em 07 de junho de 2015.

CKJM. Disponível em: http://www.spinellis.gr/sw/ckjm/. Acessado em: 22/10/2015.

CyVis. Disponível em: http://cyvis.sourceforge.net/. Acessado em: 30 de junho de 2015.

LEHMAN, M.M., RAMIL, J.F., WERNICK, P.D., PERRY, D.E., TURSKI, W.M. Metrics and Laws of Software Evolution - The Nineties View. In: **International Symposium on Software Metrics**, 4, 1997, Albuquerque, p. 20-33.

McCABE, T.J.; A Complexity Measure, Software Engineering, **IEEE Transactions on** Volume: SE-2, Issue: 4, 1976.

Metrics. Disponível em: http://sourceforge.net/projects/metrics/. Acessado em: 20 de outubro de 2015.

SOMMERVILLE, Ian. **Engenharia de Software**. 9a Ed. São Paulo: Pearson Prentice Hall, 2011.

PRESSMAN, R. S., **Engenharia de Software** - Uma Abordagem Profissional. 7ª Edição. São Paulo: McGraw Hill, 2011.

Projeto JaBUTi Metrics. Disponível em: http://code.google.com/p/jabutimetrics/. Acessado em: 20 de outubro de 2015.