ANÁLISE COMPARATIVA DO USO DE MÉTRICAS COMO FATOR DE QUALIDADE NO DESENVOLVIMENTO DE SOFTWARE

Eduardo José Magalhães[1], José do Carmo Rodrigues[2]

eduardo.magalhaes@ueg.br, jcrodrigues@uol.com.br

[1]Universidade Estadual de Goiás – Câmpus Goianésia – Sistemas de Informação
Goianésia – GO

[2] Universidade Estácio de Sá – UNESA – Engenharia e Arquitetura de Software
Rio de Janeiro - RJ

RESUMO

Este artigo tem como objetivo mostrar o método da engenharia de *software* conhecido como métricas de *software*. Essa técnica surgiu como necessidade após a crise do *software* dos anos 60, que teve entre outros motivos a complexidade no desenvolvimento de *software*. A técnica tem como princípios especificar as funções de coleta de dados de avaliação e desempenho, atribuir essas responsabilidades a toda a equipe envolvida no projeto, reunir dados de desempenho pertencentes à complementação do *software*, analisar os históricos dos projetos anteriores para determinar o efeito desses parâmetros e utilizar esses indicadores para pesar as previsões futuras. Procurou-se determinar nesse estudo as melhorias implícitas na qualidade do produto quando da adoção desse método da engenharia de *software*. A pesquisa bibliográfica compreendeu estudos das métricas LOC – *Lines of Code* (Linhas de Código) e PF – *Function Points* (Pontos de Função), assim como estudos sobre qualidade de *software* segundo institutos com reconhecimento internacional, prosseguindo com uma análise comparativa mostrando resultados obtidos em projetos utilizando métricas de *software*. Como resultado concluiu-se que com o uso de uma métrica adequada seria possível haver melhorias consideráveis na qualidade do produto, assim como em outros fatores que são determinantes e influenciam na qualidade do processo, tendo em vista reflexos na diminuição de custos, no seu controle, nos prazos de entrega, e na gerência do processo e projeto de *software*.

Palavras-Chave – Engenharia de Software, Métricas, Qualidade, Software,.

COMPARATIVE ANALYSIS OF METRICS FOR USE AS QUALITY FACTOR IN SOFTWARE DEVELOPMENT

ABSTRACT

This article aims to show the software engineering method known as software metrics. This technique has emerged as a necessity after the software crisis of the 60s, which had among other

things the complexity of software development. The technique has the principles specify the functions of collection evaluation and performance data, assign these responsibilities to all staff involved in the project, gather performance data pertaining to software completion, analyze the history of a previous project to determine the effect of these parameters and use those indicators to weigh future forecasts. It was assessed in this study implied improvements in product quality when the adoption of this method of software engineering. The bibliographic research included studies of LOC metrics - Lines of Code (Code Lines) and PF - Function Points (Function Point) as well as studies on software quality second institutes with international recognition, continuing a comparative analysis showing results in projects using software metrics. As a result it was found that with the use of a suitable metric would be possible to have considerable improvements in product quality, as well as other factors are determining and influencing the quality of the process in order reflections in the reduction of costs, in its control in terms of delivery, and process management and software design.

KEYWORDS – Metrics, Quality, Software, Software Engineering.

I. INTRODUÇÃO

O cenário atual de alta competitividade entre as organizações faz com que a necessidade da oferta de produtos e serviços diferenciados com qualidade e preços cada vez melhores, torne-se um fator crítico para a determinação do sucesso ou fracasso de uma organização. Nesse contexto, uma característica peculiar relacionada à área tecnológica pode ser destacada pelas suas rápidas mudanças, e essas mudanças atingem todo o mercado de Tecnologia da Informação (TI), inevitavelmente aumenta também as exigências do consumidor deste mercado. O desafio então passa a ser o de desenvolver produtos e serviços que satisfaçam as expectativas de qualidade de um grupo cada vez mais acostumado com a rápida evolução tecnológica a cada versão de um produto, por outro lado, as empresas procuram artifícios para que seus produtos sejam desenvolvidos no prazo estipulado com os custos estimados e com os recursos planejados.

Em todos os aspectos o fator qualidade é um objetivo a ser alcançado pelas empresas. Hoje, têm-se intensificado os estudos e tem havido mudanças no processo e no projeto de desenvolvimento de *software*, isso se dá devido à utilização de um método conhecido na engenharia de *software* como métricas de *software*. O objetivo desse método é fornecer parâmetros para auxiliar os engenheiros na tomada de decisão durante o desenvolvimento de produtos de *software.* Com a utilização desse método os *softwares* passam a ser medidos, provendo assim informações que são captadas, armazenadas e analisadas, formando um arcabouço com histórico de projetos a fim de derivar previsões de desempenho futuro. Esses atributos tornam o *software* mensurável, dando condições de dimensionar a sua qualidade, além de ser útil ao engenheiro, servindo como indicador e dando-lhe conhecimento amplo para o desenvolvimento de um projeto mesmo antes da sua existência.

Neste artigo, é apresentado um método da engenharia de *software* que permite tornar *softwares* mensuráveis, facilitando a sua compreensão, seu controle, prevendo e melhorando projetos, processos e produtos de *software*. Será feita uma abordagem das métricas LOC e PF, comumente utilizadas atualmente com maior frequência pelas empresas e órgãos do governo brasileiro, além de mostrar a contribuição no desenvolvimento de *softwares*, observando a sua influência na melhoria do processo e do produto, na qualidade e nos demais fatores como estimativas de custos, prazos e recursos. Será fator de avaliação a adoção dessa prática, seus benefícios e principais obstáculos encontrados para a utilização de métricas no processo de desenvolvimento de *software.*

II. FUNDAMENTAÇÃO TEÓRICA (1)

A. Métricas

B. Medidas, métricas e indicadores

Até os anos 70 a engenharia de *software* praticamente não existia, havia uma grande demanda por *software*, porém existiam também sérios problemas no seu desenvolvimento. Segundo Luiz (2010), a crise do *software* foi um termo cunhado para descrever as dificuldades enfrentadas no desenvolvimento de *software* no fim da década de 60. A noção de crise e as suas causas estavam ligadas inteiramente a complexidade do desenvolvimento de *software* e a pouca maturidade da engenharia de *software*. Os principais problemas que levaram a crise do *software podem* ser destacados entre outros como, estouro de orçamento nos projetos, não cumprimento de prazos,

software sem qualidade e *softwares* que muitas vezes não contemplavam os requisitos definidos. Nos últimos anos tem havido uma grande evolução na área de desenvolvimento de *software* com o surgimento de ferramentas, métodos e técnicas, no entanto, a situação não mudou conforme o que se esperava. De forma geral, ninguém poderia prever que *software* se tornaria indispensável para negócios, ciência e engenharia; que iria viabilizar a criação de novas tecnologias; que se tornaria a força motriz por trás da revolução do computador pessoal é o que enfatiza Pressman (2011, p. 30).

Em diversas áreas da engenharia a medição é algo extremamente comum. Nesse contexto Sommerville (2007, p. 432) destaca que a medição de *software* se dedica a derivar um valor numérico para algum atributo de um produto de *software* ou de um processo de *software*, porém, alguns membros da comunidade continuam a argumentar que o *software* é algo incomensurável e que tentativas de medições devem ser adiadas até haver um entendimento melhor sobre o *software* e os atributos que deverão ser usados para descrevê-lo, é o que enfatiza Pressman (2011, p. 538). Essa discordância vem do fato de que *software* não é algo que possa ser medido de forma exata, além da subjetividade podem ser somados questionamentos sobre o que medir em um *software*. Pressman (2011, p. 598) explica que o processo de medição de *software* pode ser dividido em etapas de maneira que permita um melhor entendimento das tarefas associadas à medição de um *software*.

C. Métricas e Qualidade

1. Como as métricas influem na qualidade e como qualidade depende de medições

Com base na definição da norma ISO/IEC 9126, qualidade pode ser compreendida como um agrupamento de características a serem satisfeitas de modo que o produto de *software* atenda às necessidades de seus usuários. Entretanto, esse nível de satisfação nem sempre é alcançado de forma espontânea, devendo ser continuamente construído. Para Reis (2009, p. 24), A qualidade de *software* reúne a adequação das características explícitas e implícitas. Kalinowski e Spínola (2007, p. 70) destacam que entre as atividades que podem ser realizadas para verificar a qualidade de *software* encontram-se entre outras as métricas. Atualmente garantir a qualidade de *software* é essencial, haja vista a crescente importância do *software* dentro das corporações. Apesar de tudo, é necessário entender que o problema não está somente no *software* em si, mas, sobretudo, na forma como tem sido desenvolvido, é o que afirma Morais (2010, p. 35). Contudo, para Araújo, Abreu e Mota (2010, p. 50), existem diversas formas de medir *software*, porém, destacam que utilizar métricas pode ser a forma mais fácil.

A norma internacional ISO/IEC 9126, define qualidade de software como “A totalidade de características de um produto de *software* que lhe confere a capacidade de satisfazer necessidades explícitas e implícitas”. Pressman (2011, p. 362) destaca que o padrão ISO 9126 foi desenvolvido como uma tentativa de identificar os atributos fundamentais de qualidade para *software* de computador. Nesse aspecto o padrão identifica seis atributos, a saber, funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade. Nesse contexto, a medição proporciona parâmetros que auxiliam a tomada de decisão, pois através de dados quantitativos é capaz de entender que aspectos do produto atendem ou não ao padrão de qualidade especificado.

Marques (2011) destaca que diversas são as métricas existentes e as suas aplicações no ciclo de vida de um *software*, cabe ao gerente de projeto coordenar as ações para determinar o padrão de qualidade requerido e definir quais elementos devem ser medidos e monitorados durante esse ciclo. A coleta dessas informações permite não só um melhor acompanhamento do processo de

desenvolvimento de um *software*, mas também a análise qualitativa desse *software* como um produto. A base histórica das métricas permite que futuras propostas de mudança ou criação sejam mais precisas, visto que projetos similares tendem a passar pelos mesmos problemas e soluções. Para manter ou elevar o nível de qualidade de um *software* é essencial medir e monitorar durante todo o seu ciclo de desenvolvimento. De acordo com Araújo, Abreu e Mota (2010, p. 51), "*Software* podem ser medidos (ou estimados) baseados em diversos tipos de perspectivas, como tamanho e complexidade". Observam ainda que devido às etapas do desenvolvimento diferentes métricas podem ser colhidas para o mesmo produto.

1.1 Linhas de Código (LOC) e Pontos de função (PF)

LOC é uma medida de *software* que visa estabelecer o tamanho de um sistema, baseando-se no número de linhas de código e que pode determinar o tamanho de uma aplicação já construída ou estimar o esforço a ser considerado para a obtenção de um produto a ser desenvolvido, é o que define Araújo, Abreu e Mota (2010, p. 52). De acordo com Pintaud e Oliveira (2012, p. 11), essa métrica ainda é a mais utilizada para medir o tamanho de um programa, destacam ainda que, é de fácil definição e precisão, apesar das dúvidas que muitas vezes surgem devido à interpretação da contagem do que é considerado linhas de código.

A principal vantagem desse modelo é que as linhas de código podem ser facilmente contadas por uma ferramenta automatizada. As desvantagens segundo Hazan (2009, p. 26) são principalmente relacionadas à subjetividade.

Segundo o Governo do Estado de Pernambuco (2014), a PF surgiu em 1979 como resultado de um projeto desenvolvido por Allan Albrecht, que era pesquisador da IBM. O seu objetivo era encontrar uma técnica de estimativa para esforço de desenvolvimento de *software* que fosse independente da linguagem de programação utilizada. Hazan (2009, p. 26) enfatiza que essa métrica tem sido muito utilizada, inclusive para estimar projetos dos clientes da organização do Serviço Federal de Processamento de Dados (SERPRO) desde 2003, tendo obtido segundo avaliação da autora, bons resultados com a sua aplicação.

Essa métrica é uma medida baseada nos modelos e ciclos de vida tradicionais de desenvolvimento de *software,* proposta para medir o tamanho estimado no início do desenvolvimento, é o que destaca Pintaud e Oliveira (2012, p. 12). O seu objetivo é dimensionar o *software*, quantificando a funcionalidade proporcionada ao usuário a partir do seu desenho lógico.

Outro fator que deve ser destacado, porém como vantagem da métrica PF é que ela pode ser utilizada como geradora de indicadores de recursos para estimar prazos, gerência de recursos humanos e elaboração de planos de projetos, bem como na avaliação e acompanhamento do progresso de projetos e na análise da produtividade de equipes.

III. DESENVOLVIMENTO

O artigo desenvolvido trata de uma pesquisa bibliográfica realizada no período de Julho de 2013 a Julho de 2014, por meio de consulta a livros, revistas, apostilas de estudo, dissertações de mestrado e artigos de sites da internet com relevância em engenharia de *software*. Esse estudo foi realizado utilizando como método a pesquisa bibliográfica, de natureza qualitativa que é um estudo cientifico onde é feito uma análise de forma criteriosa e ampla das publicações em uma determinada

área. A pesquisa teve como campo de ação, sobretudo a busca virtual em sites conceituados como IBM e USP, revistas com enfoque na área de engenharia de *software*, além de livros de autores conceituados internacionalmente na mesma área. Para a coleta de dados foram utilizados os seguintes descritores: engenharia de *software*, métricas de *software*, qualidade de *software*, pontos de função e linha de código.

A. Qualidade implícita do produto de *software*

Quanto ao que foi pesquisado e os passos realizados na pesquisa, ficou delimitado que o estudo estaria voltado para a melhoria implícita na qualidade do produto quando da adoção de métricas de *software*. O tema proposto sugere que o estudo em questão trate das necessidades implícitas, ou seja, a infusão desse termo propõe que as necessidades básicas dos usuários no relacionamento com o produto atualmente não são satisfeitas, e isso implica em um estudo para melhoria da qualidade do *software* nesse aspecto. É fundamental que no processo de desenvolvimento de *software* os requisitos sejam levantados com desenvoltura e solidez, pois para que haja os resultados esperados na qualidade implícita é necessário que os requisitos sejam bem definidos.

A importância no levantamento dos requisitos está associada ao conhecimento do negócio para o desenvolvimento do *software*. É nesta etapa que surgem os primeiros problemas devido o levantamento incompleto dos requisitos. O usuário conhece o negócio como um todo, está acostumado, conhece nuances, o analista por sua vez faz um levantamento de forma panorâmica com alguém que reluta em passar informações, ou pressupõe ter repassado tudo àquilo que é necessário para o bom desenvolvimento do produto de *software*. Desta forma, tem-se muitas vezes uma primeira parte do processo deficitária e ineficiente e que irá influenciar todo o processo e o produto de *software*.

B. Modelo ISO para qualidade de processo de software

Quando o termo qualidade é citado para *software,* tem-se sempre o questionamento sobre como medir a qualidade de um *software.* A tendência entre profissionais é classificar o *software* como algo imensurável, ou seja, não sendo possível a medição dos atributos de qualidade. Qualidade envolve muitos atributos de um *software*, assim não é possível medir *software* olhando apenas uma perspectiva, mas torna-se necessário o olhar sobre vários ângulos para que o produto seja completo e tenha a qualidade requerida. A norma ISO/IEC 9126 pode ser útil na definição de atributos que possibilitam enxergar fatores que são intrínsecos à qualidade do produto de *software*. O que é proposto pela norma é que os atributos de qualidade sejam distribuídos em 6 (seis) características principais e cada uma delas sendo divididas em sub-características.

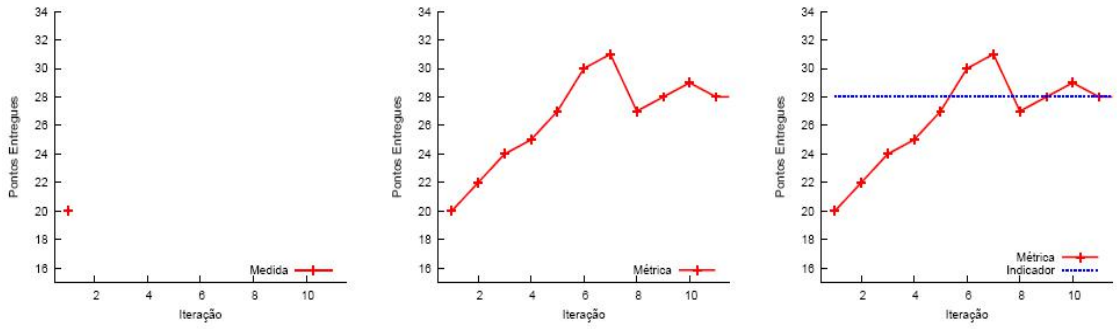
Nota-se que a principal preocupação das empresas e desenvolvedores de *softwares* está relacionada à entrega do produto requerido, pesa nesse aspecto o aumento de custos, o prazo de entrega acordado com o cliente e os demais recursos necessários. O que muitas vezes não é colocado nessa somatória é a qualidade do produto entregue. A norma 9126 sugere que cada uma das 6 (seis) características e suas sub-características componham um atributo de qualidade de *software.* Assim tem-se como atributos de qualidade a funcionalidade, a confiabilidade, usabilidade, eficiência, manutenibilidade, portabilidade e suas sub-características. Percebe-se que grande parte

das empresas que desenvolvem *software* agem da mesma forma, no andamento de um projeto aumentam o quadro de funcionários, excede as horas extras, assim como gastos com correções e erros, não conseguem cumprir os prazos, o orçamento quase sempre estoura e o produto quando é entregue não tem a qualidade esperada pelo usuário.

C. Aplicação de métricas como resultado para qualidade de *software*

Nesse aspecto torna-se fundamental para alcançar qualidade no produto a utilização do método da engenharia de *software* que consiste em medir *software* utilizando métricas. Nesse artigo é proposta para as empresas e profissionais que desenvolvem *software* a importância de medição para alcançar a qualidade esperada pelos usuários em produtos de *software*. Sendo assim, no estudo sugere-se que *softwares* sejam mensuráveis, ou seja, por meio de métodos simples, engenheiros de *softwares* comecem a coletar dados relevantes relativos aos projetos da empresa em questão. Por meio de coleta de dados, e aplicação de cálculo das métricas, sejam gerados indicadores, que são obtidos a partir da avaliação das métricas. Esses indicadores devem ser armazenados em banco de dados, formando um arcabouço ou histórico de informações de projetos da organização. Essa sugestão é apresentada na figura 1, onde são destacadas as 3 (três) fases do processo de mensuração de *software.*

Fig. 1: Total de pontos entregues por iteração



(a) Medida (b) Métrica (c) Indicador

Fonte: (Sato, 2007, p. 41)

Nesse quesito a figura 1(a) mostra a fase de coleta de dados que formam as medidas, que pode ser, por exemplo, o número de erros encontrados em determinado teste, além de outros. A figura 1(b) mostra os resultados das medições, onde os vários pontos são marcações coletadas durante um período de tempo, essas marcações formam as métricas, e a figura 1(c) mostra a interpretação das métricas, a partir dessa interpretação tem-se um indicador que pode esclarecer se algo melhorou ou piorou em um determinado projeto.

Em tese um indicador serve para nortear os gerentes de projeto com informações relevantes sobre diversas perspectivas de projetos passados apontando para projetos futuros. A fase mais importante nesse processo é a fase de medição. É nesse momento que informações são coletadas dos projetos, e devem ser feitas com cuidado e muita disciplina, pois são as informações que serão armazenadas e nortearão projetos futuramente. Desta forma, erros nesta fase podem ocasionar em

indicadores que tragam falsa indicação para os gestores de projetos e consequentemente podem comprometer o processo e produto de *software*.

É importante salientar que a utilização das métricas pode ser um fator diferencial para quem desenvolve *software,* porém, por si só não resolvem os problemas existentes no processo de desenvolvimento de *software.* É necessário, sobretudo, que a partir de um histórico de projetos sirvam para auxiliar na definição de cláusulas de contratos com clientes permitindo um melhor gerenciamento nos fatores humanos e no tempo de desenvolvimento com base no histórico da organização. Em contrapartida, quando utilizadas com maturidade podem contribuir no cumprimento de prazos, evitando muitas vezes inconvenientes no processo de desenvolvimento do produto que por vezes gera estouro nos orçamentos e garantindo uma melhoria contínua no processo e no produto de *software*.

D. Estimativas baseadas LOC

Ficou evidente no estudo que a LOC ainda é a métrica mais utilizada pelos desenvolvedores de *software*, sobretudo pela sua facilidade de utilização e medição, podendo ser feito até mesmo por ferramentas automatizadas para calcular medidas diretas. Porém, é uma métrica que contém limitações e pontos falhos na sua medição. É importante salientar que mesmo sendo uma métrica de fácil utilização por medir apenas linhas de código fonte em um determinado programa, não oferece segurança devido à maneira que cada programador escreve seus códigos, entre outros fatores também podem ser citadas as diferenças existentes entre linguagens de programação. Existem hoje inúmeras linguagens de programação e cada uma tem características particulares, as linguagens orientadas a objetos que dão suporte ao reaproveitamento de código, a tecnologia Java, por exemplo, que contém classes disponíveis para uso do programador e até mesmo a função auto completar das atuais *IDEs – Integrated development environment (Ambiente integrado para desenvolvimento de Software)* que auxiliam no desenvolvimento do código fonte, são diferenciais na produção de código.

Outro fator que deve ser destacado em contagem de linhas de código de um programa é o que deve ser contado, haja vista que quantidade de linhas de código não é sinônimo de qualidade. Ressalta-se que comentários contidos em programas para auxiliar o programador muitas vezes deixam dúvidas nessa contagem e não fica claro se devem ou não ser contados. A orientação a objetos com o conceito de reaproveitamento de código, métodos, a utilização do conceito de herança pode prejudicar um programador que utilize tal paradigma, assim como a utilização de funções recursivas. Ao contrário programadores que escrevem mais códigos e códigos menos elegantes seriam beneficiados, em contrapartida programadores mais eficientes que escrevem menos códigos para resolver um problema específico seriam prejudicados caso fosse utilizado esse método para cálculo de produtividade em uma equipe, por exemplo.

E. Estimativas baseadas em PF

A métrica PF é baseada nas medidas indiretas, portanto concentra-se na utilidade ou funcionalidade do *software,* o que torna essa métrica baseada em fatores subjetivos. Atualmente tem sido muito utilizada em comparação com a métrica LOC, empresas desenvolvedoras de *softwares* e até mesmo órgãos do governo brasileiro, entre outros, são exemplos de casos que adotaram essa

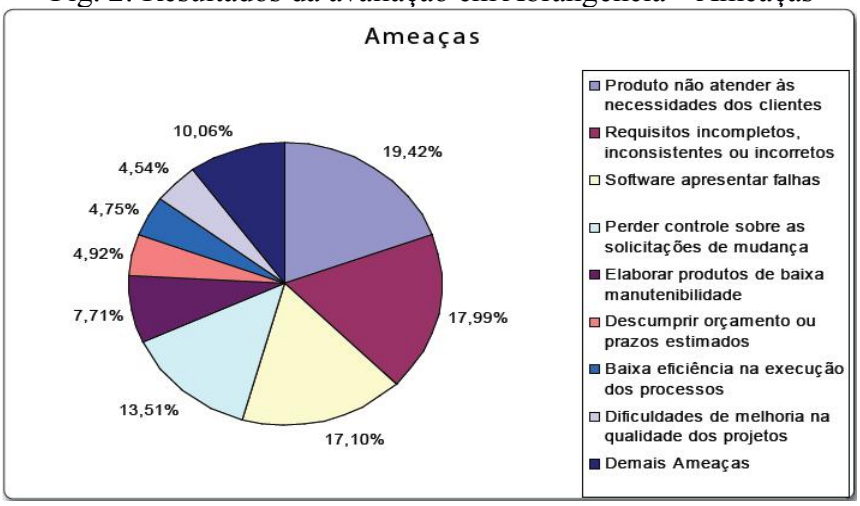
métrica. Entre os objetivos iniciais estava desenvolver uma métrica que fosse independente da metodologia de desenvolvimento e da tecnologia utilizada, desta forma, para que não haja essa interferência o fator de ponderação é baseado em medidas indiretas, como funcionalidade e manutenibilidade, por exemplo. Assim o fator subjetivo das medições indiretas tem sido alvo de questionamentos, haja vista que alguns críticos consideram a métrica apenas como uma avaliação, não tendo um resultado amplamente aceito. Por outro lado a sua recomendação é favorecida pela independência em relação à tecnologia ou modelo utilizado no desenvolvimento de *software*. Ressalta-se que com essa métrica pode-se acompanhar a evolução do produto desenvolvido, comparando resultados durante o desenvolvimento e possibilitando a sua comparação após o projeto de *software* de forma a verificar seu tamanho e custo comparando-os com o que foi originalmente definido.

É importante destacar que a PF pode medir a funcionalidade solicitada pelo usuário antes do início do projeto de *software*, tendo uma grande vantagem em relação à LOC que é necessário haver codificação para que seja feita a sua contagem. Com isso pode-se estimar tamanho e custo, sobretudo para que cláusulas contratuais sejam definidas, além de outros fatores. Para o cálculo de Pontos por Função três passos são necessários. No primeiro passo é necessário preencher uma tabela com o fator de ponderação para cada um dos parâmetros estabelecidos, e isso depende do ponto de vista da pessoa que está fazendo a análise, tem-se aqui o fator subjetivo da avaliação, onde vários fatores podem influenciar na atribuição do valor e interferir no resultado. No segundo passo é necessário avaliar questões relacionadas à funcionalidade do *software* a ser desenvolvido, o valor indicado remete ao valor de influência, que parte de nenhuma influência até o estágio essencial, variando de 0 (zero) a 5 (cinco). É importante ressaltar que para minimizar a subjetividade das medições nas etapas iniciais o IFPUG disponibiliza recomendações que podem ser seguidas. Na terceira etapa faz-se o uso de uma expressão empírica para obtenção da PF. O resultado final é um único número, cuja unidade é Pontos de Função, que mede o tamanho funcional do produto de *software.* Ademais a técnica PF fornece uma medida objetiva e comparável que auxilia a avaliação, planejamento, gerência e controle da produção de *software*.

IV. DISCUSSÃO

Um problema apontado por Pressman (2011, p. 598) na adoção das métricas de *software* é que a maioria das organizações de desenvolvimento de *software* tem menos de 20 (vinte) profissionais de *software.* Assim não é razoável, e torna-se inviável esperar que essas organizações desenvolvam programas abrangentes de métricas de *software*. No entanto, o que pode ser sugerido é que organizações de todos os tamanhos meçam e depois usem essas métricas resultantes para ajudar a melhorar o seu processo local de *software* e a qualidade e prazo dos produtos que elas produzem. É importante ressaltar que a atividade de medição implica não somente na simples existência de um repositório de padrões, somente isso não é suficiente para que *software* seja produzido com qualidade, é o que destaca Criscuolo (2007, p.22). Os fatores que determinam essas evidências podem sugerir desde ignorar ou simplesmente adotar padrões de forma desorganizada. A figura 2 mostra por meio de um gráfico os índices de ameaças em produtos de *software,* que pode ser entendido como falhas/erros em decorrência da inexistência de medição e controle no processo de *software.*

Fig. 2: Resultados da avaliação em Abrangência – Ameaças

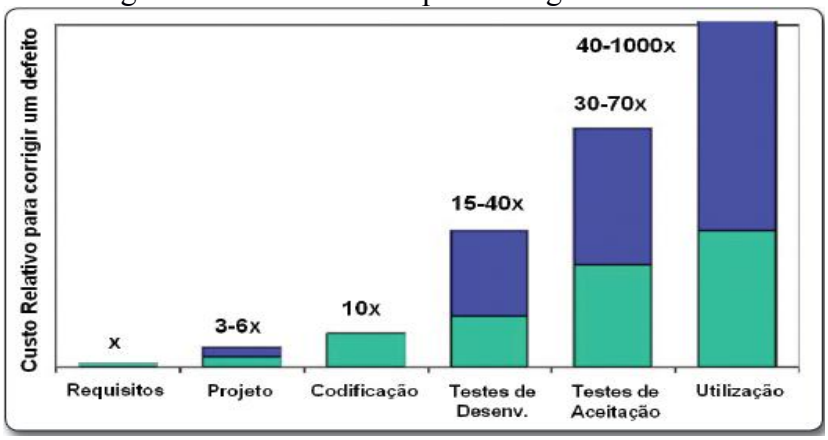


Fonte: (Espinha e Sousa, 2007, p. 21)

Quanto à qualidade do produto de *software* o maior índice indicado no gráfico (19,42%), remete ao não atendimento às necessidades dos clientes, o que evidencia que poderia haver melhorias significativas caso medições fossem adotadas no processo de desenvolvimento do *software*. O segundo maior índice (17,99%) indica falhas no início do projeto, o que pode ser demonstrado por requisitos incompletos, inconsistentes ou incorretos. Para Boegh et. al. apud Criscuolo (2007, p. 22), é necessário, portanto, que os atributos de qualidade desejados para cada projeto sejam determinados logo no início e que, com isso, seja montado um plano de qualidade.

Outro fator de grande importância para que as organizações de desenvolvimento de *software* adotem medições pode ser demonstrado pelo elevado custo para corrigir um defeito no andamento do projeto. A figura 3 mostra que quanto mais o projeto avança maior é o custo relativo para corrigir um defeito no produto, e que na etapa de utilização esse índice pode aumentar de 40 a 1000 vezes. Desta forma, com o uso de medições e o auxilio de indicadores muitos problemas da evolução de *software* poderiam ser minimizados, sobretudo questões relacionadas ao custo do *software* e ao tempo de desenvolvimento.

Figura 3 – Custo relativo para corrigir um defeito.



Fonte: Boehm apud Kalinowski e Spínola (2007, p. 70)

No caso de uma pequena organização onde se enquadra a maioria das empresas desenvolvedoras de *software* do *país,* não há uma preocupação em medir o processo de desenvolvimento de *software*, Pressman (2011, p. 598) enfatiza que essas organizações podem escolher um conjunto de medidas que podem ser obtidas facilmente, sem onerar financeiramente ou mesmo haver necessidade de grandes mudanças na estrutura da organização. Entre as medidas sugeridas estão: Esforço (homem-hora) para executar a avaliação; Tempo (horas ou dias) para fazer a alteração; Erros descobertos durante o trabalho para fazer a alteração; e, Defeitos descobertos depois que a alteração é liberada para o cliente. Outra grande vantagem na adoção de métricas pode ser percebida pela facilidade em fazer medições com o uso de ferramentas automatizadas, um exemplo é o *Plugin Metrics for Eclipse*, que é uma ferramenta gratuita e que permite o cálculo de métricas de forma automatizada.

V. CONCLUSÃO

Este artigo propôs um estudo sobre o uso de métricas como fator de qualidade no desenvolvimento de *software*. Foram apresentados tópicos que remetem à qualidade de *software* segundo parecer de institutos como ISO/IEC, além de autores consagrados internacionalmente na área de engenharia de *software*, ademais foi feita uma descrição no contexto das métricas de *software*, oportunidade onde foram detalhados os processos de coleta de dados, cálculo das métricas e avaliação das mesmas resultando em indicadores. Para exemplificar foram abordados tópicos sobre as métricas LOC e PF, entre as muitas existentes, isto feito pela facilidade em utilizar o modelo LOC e pelo uso crescente do modelo PF em empresas desenvolvedoras e órgãos do governo brasileiro, tendo essa segunda atualmente uma melhor aceitabilidade na área de desenvolvimento de *software*.

Pôde-se constatar que o método em questão pode minimizar e não eliminar totalmente os problemas existentes no processo de desenvolvimento de *software*, ante o exposto ficou evidenciado que mesmo com a sua adoção não há garantia de qualidade do *software* como produto. Porém, é necessário que haja um empenho das pessoas envolvidas no projeto, assim como organização e maturidade na avaliação e aplicação dos indicadores para que possa haver melhorias significativas. Embora o uso de métricas de *software* possa proporcionar benefícios implícitos na qualidade do produto, há ainda dificuldades para a sua ampla aceitação e uso, uma vez que grande parte das pequenas e médias organizações desenvolvedoras de *software* preferem não utilizar esse método muitas vezes por entenderem que o tempo gasto é dispendioso e sem necessidade. Em contrapartida, estudos mostraram que os *softwares* em sua grande maioria não atendem às necessidades dos clientes, além de apresentar falhas. Entretanto, fica evidenciado que o uso de métricas pode ser feito por qualquer organização de desenvolvimento de *software*, mesmo aquelas com poucos recursos e com uma quantidade pequena de desenvolvedores, isto feito sem mudanças drásticas na estrutura da empresa.

O estudo mostrou-se promissor e trouxe informações que deram maior compreensão para que problemas existentes até hoje no contexto de desenvolvimento de *software* na maioria das empresas possam ser minimizados, questões sobre economia de tempo, esforço, custo e qualidade podem ser alcançados evitando assim elevados gastos com correções de defeitos e manutenção no ciclo de vida do produto. Seria recomendável que futuramente houvesse pesquisas em empresas de

TI no Brasil, que mostrassem o índice de adoção às métricas e o percentual de melhoria no controle dos processos e na qualidade do produto segundo as práticas da engenharia de *software* comparando com o período onde não havia nenhum tipo de medição.

REFERÊNCIAS BIBLIOGRÁFICAS

ARAÚJO, Marco A. P; ABREU, Thamine C; MOTA, Leonardo S. Métricas de Software: Como utilizá-las no gerenciamento de projetos de Software. **Engenharia de Software Magazine**, Rio de Janeiro, ano 2, n. 21, p.50-55, 2010.

CRISCUOLO, Marcelo. **Qualidade de Produto de Software**: uma abordagem baseada no controle da complexidade. 2008. Dissertação (Mestrado em Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2008. Disponível em: http://www.teses.usp.br/teses/disponiveis/55/55134/tde-09052008-145857/ Acesso em: 2014-07-07.

ESPINHA, Rafael; SOUSA, J. Melhorando processos através da análise de risco e conformidade. **Engenharia de Software Magazine**, Rio de Janeiro, ano 1, nº 1, p. 10 a 21, 2007.

GOVERNO DO ESTADO DE PERNAMBUCO. Métricas de Software. 2014. Disponível em: http://www.portaisgoverno.pe.gov.br/web/metricas-de-software/definicoes Acesso em 23 de jul. 2014.

HAZAN, Claudia. Análise de Pontos de Função: Uma aplicação nas estimativas de tamanho de Projetos de Software. **Engenharia de Software Magazine**, Rio de Janeiro, ano 1, n. 02, p.25-31, 2007.

KALINOWSKI, Marcos; SPÍNOLA, Rodrigo O. Introdução à Inspeção de Software: Aumento da qualidade através de verificações intermediárias. **Engenharia de Software Magazine**, Rio de Janeiro, ano 1, n. 1, p.68-74, 2007.

LUIZ, Ricardo F. **Sem boas práticas de engenharia não há agilidade**. dez. 2010. Disponível em: https://www.ibm.com/developerworks/community/blogs/fd26864d-cb41-49cf-b719-d89c6b072893/entry/sem_boas_pr_C3_A1ticas_de_engenharia_n_C3_A3o_h_C3_A1_agilidade2?lang=en . Acesso em: 31 jul. 2014.

MARQUES, Daniela. **Métricas de Software**. São Paulo, fev. 2011. Disponível em: https://www.ibm.com/developerworks/community/blogs/tlcbr/entry/metricas_de_software?lang=en/ . Acesso em: 08 jul. 2014.

MORAIS, Lenildo. Qualidade de Software: Desvendando um requisito essencial no processo de desenvolvimento. **Engenharia de Software Magazine**, Rio de Janeiro, ano 3, n. 29, p.34-38, 2010.

PINTAUD, Marcelo F; OLIVEIRA, E. **Métricas de Desenvolvimento de Software**. São Paulo, 2012. 122p. Material Didático (Curso de pós-graduação *Lato Sensu* em Especialização em

Engenharia e Arquitetura de Software) – Universidade Estácio de Sá.

PRESSMAN, Roger S. **Engenharia de Software:** Uma abordagem Profissional. 7. Ed. São Paulo: McGraw-Hill, 2011.

REIS, Kênia. Fatores Humanos: A influência na Qualidade de Software. **Engenharia de Software Magazine**, Rio de Janeiro, ano 2, n. 18, p.24-29, 2009.

SATO, Danilo T. **Uso eficaz de métricas em métodos ágeis de desenvolvimento de software.** 2007. Dissertação (Mestrado em Ciências da Computação) – Instituto de Matemática e Estatística, Universidade de São Paulo, 2007. Disponível em: < http://www.teses.usp.br/teses/disponiveis/45/45134/tde-06092007-225914/ >. Acesso em: 2014-07-31.

SOMMERVILLE, Ian. **Engenharia de Software**. 8. ed. São Paulo: Addison-Wesley, 2007.