

Uso de métricas de software na quantificação dos custos de erros encontrados em um projeto real

Thiago Pelizoni
André Terceiro
Rodrigo Raminelli
Pedro A. Saraiva Junior

2013

Resumo

O custo de defeitos em software tende a crescer exponencialmente a medida que as etapas do desenvolvimento de software avançam. O valor pode ser alto ao ponto de custar entre 90 a 880 vezes o valor do levantamento deste requisito na sua fase inicial.(NISL, 2002, p. 37) A Engenharia de Software utiliza métricas de produto afim de construir um software de alta qualidade. Desta forma, este trabalho tem o intuito de demonstrar a aplicação das técnicas de métricas de software no projeto de inscrição dos participantes da Corrida de São Silvestre afim de quantificar os erros encontrados, seja na fase de testes, homologação ou produção e calcular o custo destes erros, evidenciando o quanto a empresa economizaria em uma melhoria no processo.

1 Introdução

São comuns abordagens a respeito de qualidade de software, que é necessário seguir boas práticas para se obter um software de alta qualidade, no entanto, como pode ser definida a qualidade? Em um sentido geral, qualidade de software é a satisfação de requisitos funcionais e de desempenho explicitamente declarados, normas de software explicitamente documentadas e características implícitas que são esperadas em todo o software desenvolvido profissionalmente. A partir desta definição, podem ser enfatizados três pontos importantes:

1. Requisitos de software são a fundação a partir da qual a qualidade é medida. A falta de conformidade com os requisitos é falta de qualidade.
2. Normas especificadas definem um conjunto de critérios de desenvolvimento que guiam o modo pelo qual o software é construído. Se os critérios não são seguidos, quase sempre, irá resultar em falta de qualidade.
3. Há um conjunto de requisitos implícitos que frequentemente não são mencionados, por exemplo, o desejo de facilidade de uso (usabilidade). Se o software satisfaz os requisitos explícitos, mas deixa de satisfazer requisitos implícitos, a qualidade do software é suspeita.

(PRESSMAN, 2006, p.349)

1.1 Métricas de Software

A qualidade de um produto é função de quanto ele muda o mundo para melhor

Tom DeMarco(PRESSMAN, 2006, p.350)

Um elemento-chave que a engenharia utiliza em seu processo é a medição, de modo a ter um melhor entendimento dos atributos dos modelos criados afim de avaliar a qualidade seus produtos ou sistemas desenvolvidos.

De acordo com Pressman, a medição é um processo pelo qual números ou símbolos são associados aos atributos de entidades do mundo real, de modo que os determinem de acordo com regras claramente definidas. Nas ciências físicas, medicina, economia e mais recentemente nas ciências sociais, atualmente é possível medir atributos que anteriormente não eram mensuráveis. Obviamente, tais medições em engenharia de software não são tão refinadas quanto muitas medições nas ciências físicas mas, elas existem e decisões importantes são tomadas com base nelas. A obrigação de tentar medir o não mensurável afim de melhorar o entendimento de entidades particulares é tão potente em engenharia de software como em qualquer outra disciplina.(PRESSMAN, 2006, p.348)

1.1.1 Medidas, Métricas e Indicadores

Uma ciência é tão madura quanto seus instrumentos de medição

Louis Pasteur(PRESSMAN, 2006, p.352)

Em engenharia de software, uma medida fornece uma indicação quantitativa da extensão, qualidade, dimensão, capacidade ou tamanho de algum produto ou processo, deste modo, podemos definir como *medição* o ato de determinar uma medida.(PRESSMAN, 2006, p.348)

Quando um ponto de dados é coletado, por exemplo, o número de erros descoberto em um componente de software, uma medida é estabelecida. Neste caso, a medição ocorre como o resultado da coleção de um ou mais ponto de dados, por exemplo, um certo número de revisões de componentes e testes de unidade são investigados afim de coletar medidas do número de erros de cada um.

Uma métrica de software visa relacionar as medidas individuais de algum modo, por exemplo, o número de erros encontrados por revisão ou o número médio de erros encontrados por teste de unidade. Com posse dessas informações, um engenheiro de software desenvolve métricas de modo que os indicadores possam ser obtidos. Um *indicador* consiste em um *métrica* ou em uma combinação de *métricas* cujo intuito é fornecer uma profundidade na visão do processo de software, projeto de software ou o produto em si.(PRESSMAN, 2006)

1.1.2 Princípios de Medição

De acordo com Pressman um processo de medição pode ser caracterizado por cinco atividades:

1. *Formulação*: A derivação de medidas e métricas de software adequadas para a representação do software que está sendo considerado.
2. *Coleta*: Mecanismo usado para acumular os dados necessários para derivar as métricas formuladas.

3. *Análise*: Cálculo de métricas e aplicação das ferramentas matemáticas.
4. *Interpretação*: Avaliação das métricas em um esforço para ganhar profundidade na visão da qualidade da representação.
5. *Realimentação*: Recomendações derivadas da interpretação das métricas de produto transmitidas à equipe de software.

(PRESSMAN, 2006, p.353)

1.2 Objetivo deste trabalho

Tendo por base as informações anteriormente citadas, o objetivo deste trabalho demonstrar a aplicação das técnicas de métricas no projeto de inscrição dos participantes da Corrida de São Silvestre, afim de quantificar os erros encontrados, sejam estes na fase de testes, homologação ou produção; calcular o custo destes erros afim evidenciar o quanto a empresa economizaria em uma melhoria no processo, detectando os erros em fases do processo de desenvolvimento em que tal detecção é menos onerosa em termos financeiros.

2 Materiais e métodos

Nesta sessão iremos descrever quais foram as tecnologias utilizadas no desenvolvimento do projeto, o processo de testes e homologação, qual o procedimento utilizado para reportar erros e consertá-los.

2.1 O projeto

O projeto ao qual foi estudo de caso descrito neste artigo foi o de inscrição dos participantes da corrida de São Silvestre 2013 desenvolvido pela Fundação Casper Líbero. Este serviu como estudo de caso conta com mais de 150.000 linhas de código. A linguagem de programação utilizada é PHP¹ na versão 5.3.18, Zend Framework 1.12², banco de dados Oracle, controlador de versão Git³ e o Trello⁴ para gerenciamento das atividades.

O projeto citado havia sido criado em 2012 porém, passou por diversas modificações arquiteturais, visuais e melhorias em geral de modo a englobar novas regras de negócio bem como, correções de problemas identificado na versão anterior.

2.2 Testes

O projeto não conta com testes unitários. Os testes em sua maioria são efetuado pelos analistas de negócio do produto de maneira manual. Afim automatizar estes testes manuais, uma bateria de testes do sistema foram criados utilizando o Selenium IDE⁵. Desta forma, a cada nova funcionalidade criada ou, correção efetuada, estes testes são rodados pelo desenvolvedor. Todavia, os responsáveis pela homologação efetuam os testes manualmente.

¹PHP é uma linguagem de programação interpretada, livre, usada originalmente apenas para o desenvolvimento de aplicações server-side, capazes de gerar conteúdo dinâmico na Web.

²Zend Framework é um framework para aplicações Web de código aberto implementado em PHP 5 completamente orientado a objetos, licenciado sob a New BSD License.

³Git é um sistema de controle de versão distribuído projetado e desenvolvido por Linus Torvalds.

⁴Trello é um organizador de tarefas e eventos bastante dinâmico e funcional, bastante utilizado para gerenciar projetos com a metodologia Scrum.

⁵Selenium IDE é um ambiente integrado de desenvolvimento para scripts de testes automatizados simulando o comportamento de um usuário.

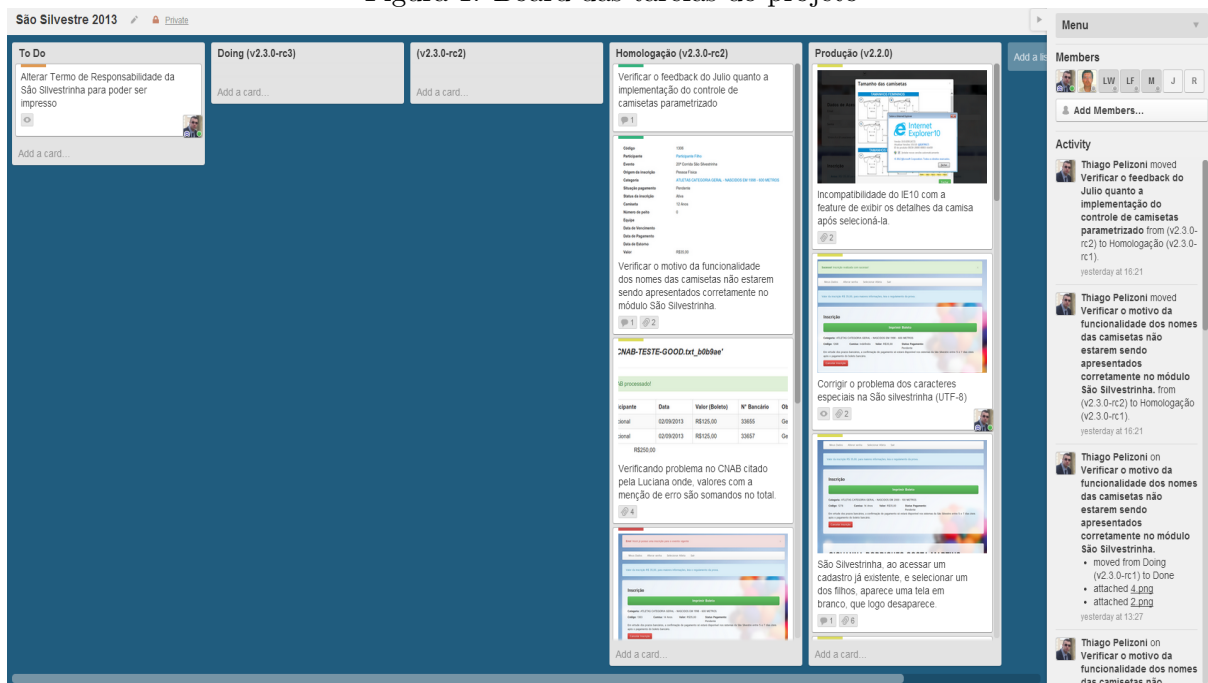
2.3 Gerenciamento do processo de desenvolvimento

A empresa não segue uma metodologia de desenvolvimento de mercado, por exemplo, Scrum, RUP, etc. Trata-se de um desenvolvimento incremental onde, cada funcionalidade é especificada, desenvolvida, testada e colocada em produção.

2.3.1 Board do projeto

Como dito anteriormente, a empresa utiliza-se do Trello para o gerenciamento das atividades de um projeto. Cada projeto refere-se única e exclusivamente a um board⁶, de modo a não misturar informações com outros projetos. Na figura a seguir podemos observar o board do projeto mencionado por este artigo.

Figura 1: Board das tarefas do projeto



Cada coluna do board representa um estágio da tarefa onde, este board está dividido em cinco colunas, sendo elas:

- *To do*: Refere-se a todas as tarefas que necessitam ser feitas. Primeiras tarefas da coluna possuem maior prioridade.
- *Doing*: Refere-se as tarefas que estão sendo feitas em um determinado momento. Cada programador pode ter apenas uma tarefa sendo feito em um dado momento. Quando algum item com maior prioridade, como por exemplo, um defeito em produção, o programador coloca sua tarefa novamente na coluna "To do", traz a tarefa que diz respeito ao problema para a coluna "Doing" e realiza a correção.
- *Done*: Refere-se as tarefas que foram concluídas. Estas tarefas, após testadas pelo desenvolvedor e evidenciadas que estão de acordo com o especificado, estarão aptas para serem homologadas por quem solicitou tal tarefa.

⁶O board de um projeto no Trello é o equivalente a Kanban online, conforme pode ser observado na figura "Board das tarefas do projeto".

- *Homologação*: Referem-se as tarefas que encontram-se em ambiente de homologação sendo analisadas pelos solicitantes. Caso haja um problema, uma tarefa com o rótulo de "Correção" será criado na coluna "To do", e passará pelo processo descrito anteriormente. Caso as tarefas estejam de acordo, estas estarão áptas para irem para a produção.
- *Produção*: Referem-se as tarefas que foram homologadas pelos solicitantes, geralmente, os analistas de negócios. Quando um problema em produção é encontrado, a tarefa criada faz referência a tarefa na coluna de produção, afim de manter um histórico.

2.3.2 Tarefas

Toda e qualquer nova tarefa, seja uma nova funcionalidade, alteração ou correção de bugs encontrados, independente do ambiente ao qual este se encontra, é gerado um card⁷ no Trello.

⁷Um card nada mais é do que uma tarefa dentro do board do Trello.

Figura 2: Uma tarefa do projeto

Comportamento inadequado no processamento do CNAB.
in list **Produção (v2.2.0)**

Boa tarde,
[redacted], conforme falamos, favor verificar o ret em anexo.... o sistema não deu pagamento na inscrição do atleta, mas na tela de consulta ao CNAB, o valor de R\$1,25, soma-se ao total de entradas.
Abaixo, a linha de erro.....
123330 [redacted]

R\$125,00
106694
Gerado com erro
Valores divergentes: CNAB R\$1,25, Inscrição/Pagamento R\$125,00.
Nº Pagamento = 106734. Nosso Numero = 106694.

[Edit the card description.](#)

Attachments [Attach](#)

TXT **CNAB-TESTE.txt**
Oct 23 at 11:13 - 1.57 KB
[Delete](#)

CNAB-fixed.png
Oct 23 at 11:13 - 46.75 KB
[Remove Cover](#) - [Delete](#)

Checklist

100%

- ☒ Criar um arquivo de CNAB afim de reproduzir o comportamento citado
- ☒ Identificar ponto da aplicação responsável por tal comportamento
- ☒ Corrigir
- ☒ Testar

[Add item](#)

Labels
Bug
[Edit Labels...](#)

Members

[Assign...](#)

Actions
[Add checklist...](#)
[Due date...](#)
[Attach File...](#)
[Move...](#)
[Subscribe](#)
[Archive](#)

Card #29 [More...](#)

Como pode ser observado acima, uma tarefa segue uma estrutura básica:

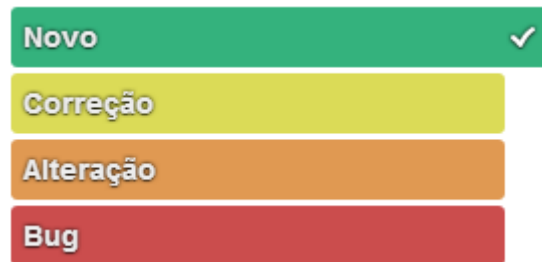
- **Título:** Uma breve descrição da tarefa a ser realizada.
- **Descrição:** Uma descrição detalhada da tarefa a ser realizada.
- **Rótulo:** Qual é o tipo da tarefa.
- **Membros:** Quem é o responsável por esta tarefa. Pode haver mais de uma pessoa dependendo do caso, por exemplo, um desenvolvedor e um analista de negócios.
- **Checklist:** Quais os passos necessários para a conclusão da tarefa, afim de mensurar em quantos por cento esta atividade se encontra quando em execução.

- *Anexos*: Evidências de que a funcionalidade ou as correções estão de acordo com o especificado. Em grande parte dos casos são screenshots.

2.3.3 Rótulo para as tarefas

Cada tarefa possui um rótulo, de modo a ficar mais fácil identificar o que cada tarefa é através de cores. Desta forma, ao olhar o board de modo geral, através dessas cores é possível identificar e reordenar as tarefas de acordo com sua prioridade de maneira mais fácil.

Figura 3: Rótulos das tarefas



Como pode ser observado na imagem acima, atualmente, as tarefas podem ser de quatro tipos diferentes:

- *Novo*: Refere-se a alguma nova funcionalidade na aplicação.
- *Correção*: Refere-se a correção de algum problema identificado no processo de homologação de uma funcionalidade, por exemplo, se uma funcionalidade foi implementada porém, ao verificá-la constatou-se um problema, um card com este rótulo será aberto para a devida correção.
- *Alteração*: Refere-se a alteração de alguma funcionalidade na aplicação.
- *Bug*: Refere-se a um problema encontrado em ambiente de produção. Geralmente estas tarefas em mais prioridade do que as demais.

2.4 Indicadores

Conforme descrito no processo de desenvolvimento, todas as informações referente ao gerenciamento do projeto encontram-se na ferramenta Trello. Todavia, há alguns meses, o responsável pela equipe de desenvolvimento foi desligado da empresa e, uma enorme quantidade de informações do projeto São Silvestre foi perdido contendo toda a base histórica do projeto do ano de 2012 e uma boa quantidade das informações do projeto de 2013 pois, o board do projeto estava atrelado a conta pessoal deste funcionário e não da empresa.

Uma tarefa pode conter um ou vários commits para sua conclusão. Isto muitas vezes deve-se a complexidade da mesma. Para toda e qualquer tarefa, um novo *branch* é criado afim de isolar ela da linha principal enquanto tal funcionalidade não estiver homologada afim de fazer parte da linha principal da aplicação, o *branch master*.

Desta forma, além do board do projeto São Silvestre atual, o histórico de commits do repositório do projeto no BitBucket⁸ será utilizado afim de levantar informações precisas para este trabalho, desde de o começo do desenvolvimento deste até sua versão mais atual **2.4.0**.

⁸BitBucket é um serviço gratuito de hospedagem de projetos de software que utilizam Git ou Mercurial como controle de versão. <https://bitbucket.org/>

3 Resultados

3.1 Goal Question Metrics

A técnica *Goal Question Metrics* consiste em um método simples para planejar medições de forma que estas sejam baseadas em objetivos específicos. Portanto, iremos descrever quais foram os objetivos dessas medições bem como, iremos discorrer a respeito dos resultados encontrados.

3.2 Objetivo

Analisar as estimativas do projeto de inscrição dos participantes da Corrida de São Silvestre com o propósito de quantificar todos os defeitos encontrados, qualificá-los de acordo com a etapa do processo, seja em homologação ou produção, com respeito a quantificar as horas gastas para corrigir o problema, qualificar essas horas de acordo com os profissionais envolvidos de modo a descobrir o total gasto com estes problemas a fim de evidenciar o quanto um investimento em uma melhoria do processo de desenvolvimento, a partir da aplicação de testes unitários e integração contínua seria econômico para a empresa no contexto de desenvolvimento de software.

3.3 Questão 1

Qual é o número de incidências encontradas desde o início do projeto até a versão mais atual?

3.3.1 Resposta

Através de uma minuciosa análise do repositório do projeto, entre o período de 17/04/2012 até 13/11/2013 foram encontrado 176 incidências categorizadas como *alterações*, *correções* e *bugs*, estando eles divididos da seguinte forma:

- **Alterações:** 39 incidências encontradas
- **Correções:** 98 incidências encontradas
- **Bugs:** 98 incidências encontradas

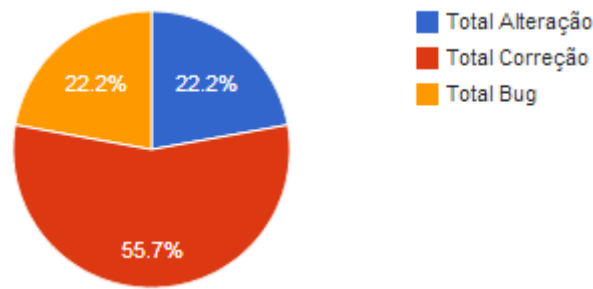
3.4 Questão 2

Qual é o percentual de cada incidência de acordo com a sua categoria?

3.4.1 Resposta 2

A partir dessas informações, foi criado um gráfico de modo a visualizar qual é a porcentagem de cada categoria de incidências encontradas:

Figura 4: Métricas encontradas



(PELIZONI, 2013)

3.5 Questão 3

Qual é o número mínimo de profissionais envolvidos para a resolução de cada incidência

3.5.1 Resposta 3

O número mínimo de profissionais para a resolução cada incidência, independente qual seu tipo, será de 3 (três) profissionais, sendo eles:

- *Programador*: Profissional que realizará a tarefa de correção da incidência encontrada.
- *Analista / Testador*: Profissional que irá homologar se a tarefa foi realizada da maneira correta.
- *Administrador de Sistemas*: Profissional responsável por atualizar tanto o ambiente de homologação quanto de produção.

3.6 Questão 4

Qual é o número de horas gasta por cada profissional envolvido no processo para a resolução de uma incidência?

3.6.1 Resposta 4

Cada incidência, independente de seu tipo, pode carregar em si uma intrínseca complexidade pois, em muitos casos, os fatores ou dados necessários para a reprodução do problema são mais complexos de se obter ou simular do que muitas vezes corrigir o problema propriamente dito.

O programador, com base nas informações de uma incidência, reunir dados para a reprodução, corrigir e evidenciar que essas correções estão de acordo com o comportamento adequado para a funcionalidade proposta.

O analista ou testador, a partir dos dados disponíveis para a reprodução do problema irá verificar a correção realizada, em caso de aprovação, irá para o ambiente de produção, caso contrário, uma nova tarefa será criada para corrigi-la.

O administrador de sistemas, irá atualizar ambos os ambientes, ou seja, quando o programador disponibilizar a correção, esta primeiramente será aplicada em ambiente de homologação e, posteriormente, em produção.

Deste modo, com base em uma média de base histórica, a estimativa para cada profissional será:

- **Programador:**
 - Alteração: 7 horas
 - Correção: 8 horas
 - Bug: 12 horas
- **Analista / Testador:** 3 horas
- **Administrador de Sistemas:** 2 horas

3.7 Questão 5

Qual é o valor da hora de trabalho de cada profissional envolvido no processo de resolução de uma incidência?

3.7.1 Resposta 5

De modo a calcular o valor destes profissionais, será utilizado a média salarial entre 2012 e 2013 pago pela empresa responsável pelo projeto. O método utilizado para se obter o valor hora de cada profissional é resultado da seguinte fórmula:

$$\text{Valor Hora} = \frac{\text{Valor Base}/30}{8}$$

Desta forma, temos a partir da média do salário base o valor hora de cada profissional:

- **Programador:** Salário base: R\$ 4.300,00, salário hora: R\$ 17,92.
- **Analista / Testador:** Salário base: R\$ 5.500,00, salário hora: R\$ 22,92.
- **Administrador de Sistemas:** Salário base: R\$ 6.700,00, salário hora: R\$ 27,92.

3.8 Questão 6

Qual é a estimativa de horas gastas para a resolução de uma incidência encontrada?

3.8.1 Resposta 6

A estimativa de horas gasta por uma incidência é resultado da soma das horas trabalhadas por cada profissional envolvido no processo, sendo assim temos:

- Alteração: 12 horas
- Correção: 13 horas
- Bug: 17 horas

3.9 Questão 7

Qual é a estimativa do total de horas gastas para a resolução das incidências encontradas?

3.9.1 Resposta 7

De acordo com o número de horas estimado por incidência, podemos estimar que, para as 39 incidências de alterações foram gastas 468 horas, para as 98 incidências de alterações foram gastas 1274 horas e para as 39 incidências de 98 foram gastas 663 horas para suas devidas correções.

Portanto, o total de horas gasta com todas as incidências encontradas foi de 2405 horas.

3.10 Questão 8

Qual é a estimativa do valor gasto para a resolução de uma incidência encontrada?

3.10.1 Resposta 8

A estimativa do valor gasto por uma incidência é resultado da soma do valor hora de cada profissional envolvido no processo, sendo assim, uma incidência específica é resultante da multiplicação do número de horas gastos por cada profissional com seu respectivo valor hora, sendo assim temos:

- Alteração: $(7 * R\$ 17,92) + (3 * R\$ 22,92) * (2 * R\$ 27,92) = R\$ 250,04$.
- Correção: $(8 * R\$ 17,92) + (3 * R\$ 22,92) * (2 * R\$ 27,92) = R\$ 267,96$.
- Bug: $(12 * R\$ 17,92) + (3 * R\$ 22,92) * (2 * R\$ 27,92) = R\$ 339,64$.

3.11 Questão 9

Qual é a estimativa do valor total gastos para a resolução das incidências encontradas?

3.11.1 Resposta 9

De acordo com o valor unitário estimado por incidência estimamos que, para as 39 incidências de alterações foram gastos $R\$ 9,751,56$ reais, para as 98 incidências de correções foram gastos $R\$ 26.260,08$ reais e para as 39 incidências de bugs foram gastos $R\$ 13.245,96$ reais.

Portanto, o valor total gasta com todas as incidências encontradas foi de $R\$ 49.257,60$ reais.

4 Conclusão

Com base nos fatos citados, podemos observar que o alto número de incidências encontradas através de análise do repositório, bem como do board de gerenciamento do projeto, evidencia que o uso de uma ferramenta como bugtracker seria mais eficaz para gerenciar este tipo de informação.

O custo dos bugs e correções somados é equivalente a 4 meses de salário bruto de um programador da empresa no período citado.

- Avaliar a forma como as medidas foram extraídas e comentar sobre as vantagens (mais rápido, mais preciso) e desvantagens (treinamento inicial no uso da ferramenta) de usar um bugtracker em vez de extrair as medidas analisando o repositório. - Foi criada uma base para comparação: agora é possível em outros projetos da FCL ter uma base de comparação, por exemplo: alterações/kloc, correções/kloc, bugs/kloc. Então, nos projetos seguintes pode ser medido também o número de bugs/kloc e comparado com o valor da São Silvestre; - Do do total de problemas (138 = correções

+ bugs), 28,2% são detectados em produção: $(\text{bugs}) / (\text{correções} + \text{bugs}) = 39 / 138 = .282$. Independente de ser possível aplicar ou não na FCL, isto indica que deve-se pensar em montar uma equipe independente de testes ou no mínimo treinar melhor as pessoas que fazem os testes atualmente.

4.1 Trabalhos Futuros

- Rascunhar brevemente um cálculo de como o Jenkins compensaria para a empresa (independente se questões internas da FCL deixam a aplicação disto na prática), indicando que isto pode ser analisado com mais detalhes em um trabalho futuro. Este "rascunho breve", no meu ponto de vista, não seria complexo. O trabalho do administrador de sistemas, pelo que entendi, é de um Jenkins manual. Ele leva 2 horas para fazer. Alguém apertar um botão no Jenkins levaria 5 minutos (precisaríamos conseguir evidências deste tempo por referências bibliográficas). Logo economizamos 115 minutos por ocorrência. Podemos citar "por cima" os investimentos: instalação do Jenkins, treinamento etc, configuração em cada projeto, indicando que é promissor e que em outro trabalho pode ser feito um estudo mais aprofundado; - Comentar (sem detalhar) sobre a comparação do número de correções, bugs e alterações com valores do mercado ou da literatura, para avaliar se os números da FCL estão melhores, iguais ou piores. Podemos citar que a comparação via bugs/kloc, alterações/kloc, correções/kloc, para projetos com grandes diferenças (linguagem de programação, desktop x web etc) pode gerar resultados imprecisos (LLC, 2013a) (MURDICO, 2007) e que um estudo cuidadoso deve ser feito para chegar-se a um valor que realmente sirva como base para comparação; - Comentar que um trabalho futuro poderia quantificar o quanto um investimento em mais testes evitaria um custo maior de erros em produção (LLC, 2013b) - Pode-se detalhar os motivos das correções (itens não especificados? Erro do desenvolvedor? Ambiguidade na especificação?) para saber melhor futuramente em que pontos pode-se agir para colher mais resultados

Referências

- LLC, C. J. . A. *A short history of the lines of code metric*. 2013. URL: Disponível em: <http://namcookanalytics.com/wp-content/uploads/2013/07/LinesofCode2013.pdf> (Acesso em 27 novembro 2013).
- LLC, C. J. . A. *Software defect origin and removal methods*. 2013. URL: Disponível em: <http://namcookanalytics.com/wp-content/uploads/2013/08/Software-Defect-Origins-and-Removal-Methods2013.pdf> (Acesso em 27 novembro 2013).
- MURDICO, V. *Bugs per lines of code*. 2007. URL: Disponível em: <http://amartester.blogspot.com.br/2007/04/bugs-per-lines-of-code.html> (Acesso em 27 novembro 2013).
- NISL. *The Economic Impacts of Inadequate Infrastructure for Software Testing - Final Report*. 2002. URL: Disponível em: <http://www.nist.gov/director/planning/upload/report02-3.pdf> (Acesso em 24 outubro 2013).
- PELIZONI, T. *Relatório de Métricas do projeto São Silvestre*. 2013. URL: Disponível em: https://docs.google.com/spreadsheet/ccc?key=0AnYaG_iUd-LrdDhXMDILbWd5WVlxdUR3MVBLCjFZR2c&usp=sharing#gid=0.pdf (Acesso em 14 novembro 2013).
- PRESSMAN, R. S. *Engenharia de Software*. Reading, Massachusetts: Mcgraw-hill Interamericana, 2006.