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 os 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 Sao Silvestre 2013 desenvolvido pela Fundação Casper Líbero. Este serviu como estudo de caso conta com mais de 150.000 linhas de de código. A linguagem de programação utilizada é PHP[1] na versão 5.3.18, Zend Framework 1.12[2], banco de dados Oracle[3], controlador de versão Git[4] e o Trello[5] 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 IDE[6]. Desta forma, a cada nova funcionalidade criada

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

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

[3]Oracle é um SGBD (sistema gerenciador de banco de dados)

[4]Git é um sistema de controle de versão distribuído projetado e desenvolvido por Linus Torvalds.

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

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

ou, correção efetuada, estes testes são rodados pelo desenvolvedor. Todavia, os responsáveis pela homologação efetuam os testes manualmente.

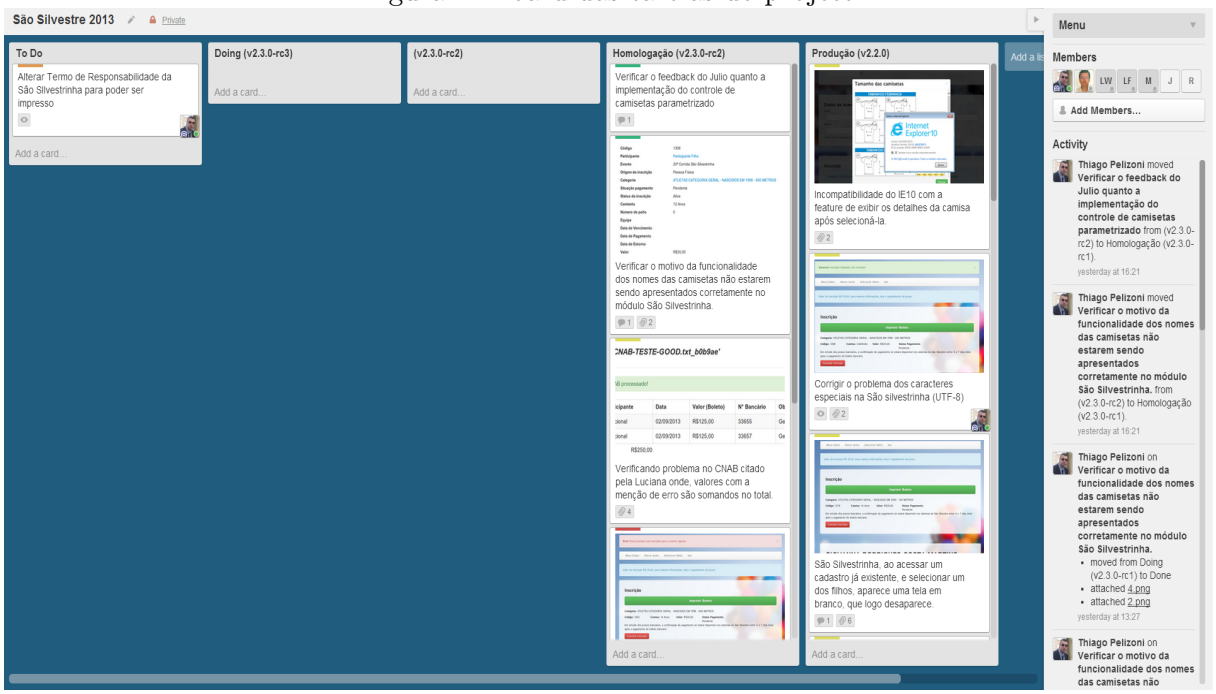
2.3 Gerenciamento do processo de desenvolvimento

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 unica e exclusivamente a um board[7], 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.

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

- *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.
- *Homologação*: Referem-se as tarefas que encontram-se em ambiente de homologação sendo analizadas 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 card[8] no Trello.

[8]Um card nada mais é do que uma tarefa dentro do board do Trello.

Figura 2: Uma tarefa do projeto

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

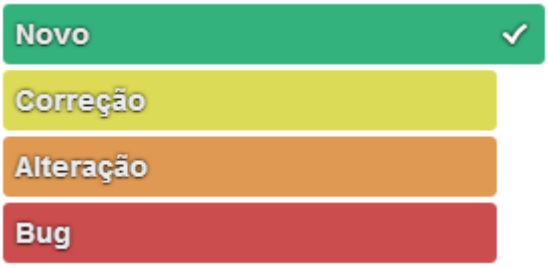
- *Título*: Uma breve descrição da tarefa a ser realizada.
- *Descrição*: Uma descrição detalhada da tarefa a ser realizada.
- *Rótulo*: Qual é o tipo da tarefa.
- *Membros*: Quem é o responsável por esta tarefa. Pode haver mais de uma pessoa dependendo do caso, por exemplo, um desenvolvedor e um analista 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.

Desta forma, além do board do projeto São Silvestre atual, o histórico de commits do repositório do projeto no BitBucket[9] será utilizado afim de levantar mais informações para este trabalho, desde de sua versão **1.0.0** até sua versão mais atual, **2.4.0**.

Portanto, o intuito deste tratalho é fazer uma minuciosa investigação afim de quantificar todos os defeitos encontrados, qualificá-los de acordo com a etapa do processo, seja em homologação ou produção, 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

[9]BitBucket é um serviço gratuito de hospedagem de projetos de software que utilizam Git ou Mercurial como controle de versão. https://bitbucket.org/

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.

3 Resultados

Resultados, o que foi observado

4 Conclusão

Considerações finais, o que se pode concluir dos resultados?

Referências

NISL. *The Economic Impacts of Inadequate Infrastructure for Software Testing - Final Report*. 2002. URL: http://www.nist.gov/director/planning/upload/report02-3.pdf (Acessado em 10/2013).

PRESSMAN, R. S. *Engenharia de Software*. Reading, Massachusetts: Mcgraw-hill Interamericana, 2006.