Projeto Final da Disciplina de Requisitos de Software

Ferramenta de Gerência de Requisitos

Requisitos de Software

Rafael Fazzolino Pinto Barbosa - 11/0136942
Thiago Ramires Kairala - 12/00****
Eduardo Brasil Martins - 11/0115104
Bruno Contessotto Bragança - 11/09*****

Brasília, DF - 2014

Histórico de Alterações

Sumário

1 Introdução

1 .1 Contexto

O desenvolvimento de software perpassa inúmeras fases até que o mesmo seja concluído e entregue ao cliente, uma dessas fases, e provavelmente a mais importante é a fase em que devemos entender o problema do usuário, compreender exatamente o que o mesmo necessita e apresentá-lo uma solução. Nesta fase, negociações serão feitas, tanto sobre funcionalidades do sistema quanto custo do projeto, tempo para conclusão do mesmo e restrições de projeto.

O resultado desta fase é uma documentação robusta, principalmente ao utilizar metodologias tradicionais de desenvolvimento. Nesta documentação se encontram as funcionalidades do software, as características do mesmo e as restrições de projeto, podendo abranger todo o software ou apenas uma primeira etapa de desenvolvimento, como é feito em metodologias ágeis, estes ítens são conhecidos como requisitos.

Com esta documentação em mãos, os desenvolvedores podem começar o desenvolvimento do sistema, porém muitos problemas surgem tanto na construção da documentação quanto na utilização da mesma para o desenvolvimento do software. A gerência, organização, classificação e rastreabilidade dos requisitos geram inúmeros problemas, já que o conteúdo é muito extenso para ser organizado facilmente de forma adequada. A partir deste problema, surge a necessidade da utilização de ferramentas que auxiliem na organização, classificação e rastreabilidade dos requisitos.

Inúmeras ferramentas já foram criadas para esta finalidade, contudo as ferramentas gratuitas não possuem a qualidade necessária para desenvolvimento de grandes e críticos sistemas. A partir daí, surge a necessidade da criação de uma nova ferramenta, de código aberto, grats e com qualidade suficiente para suprir as necessidades de todos os desenvolvedores de software do mundo.

1 .2 Objetivos

Objetivos Gerais

[APRESENTAR OS OBJETIVOS GERAIS DO TRABALHO].

Objetivos Específicos

- Entender e registrar o problema do cliente.
- Entender e registrar as necessidades do cliente.
- Definir requisitos funcionais para o desenvolvimento da ferramenta.
- Definir requisitos não-funcionais para o desenvolvimento da ferramenta.
- Definir restrições de projeto.
- Definir escopo e custo do projeto.

1 .3 Justificativas

[APRESENTAR AQUI A JUSTIFICATIVA DA ESCOLHA DO TEMA DA DISCIPLINA. PRA QUE ESTAMOS FAZENDO ESSE TRABALHO?]

2 Fundamentação teórica

[SEGUE UM EXEMPLO DE FUNDAMENTAÇÃO TEÓRICA USADO NO TRABALHO DE MEDIÇÃO E ANÁLISE]

Há alguns bons motivos para coletar e rastrear métricas, quando bem utilizadas. Métricas são interessantes para mensurar o progresso do código, porém podem ser vistas como anti-éticas para mensurar a performance de um membro em especial [1].

As métricas são importantes para rastrear motivos de, por exemplo, a cobertura caiu de 80% pra 60%, ou qual o motivo do *velocity* do time decaiu em determinada semana. Além de que, com as métricas é possível com o decorrer do tempo estimar e prever se a equipe terá chances de concluir determinado projeto em tempo, servindo assim como *feedback* para a equipe e vindo a servir de motivação para a o desenvolvimento

> *"There are three kinds of lies: lies, damned lies, and statistics."*
> Benjamin Disraeli

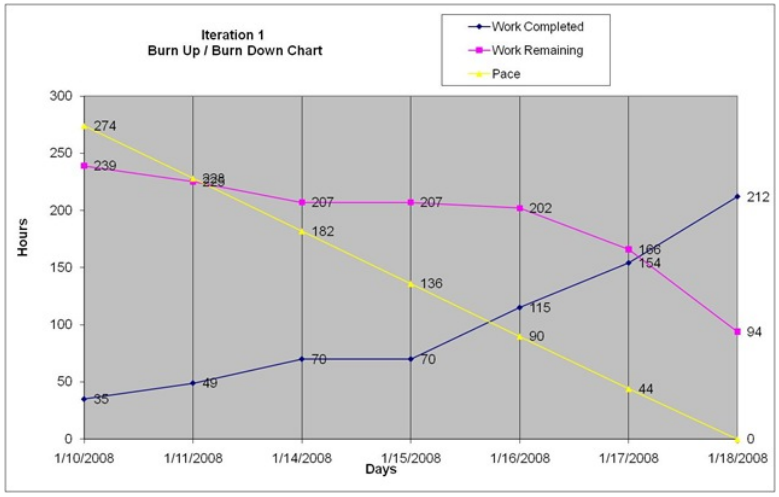

Figura 1. Exemplo de gráfico de *Burn up & Burn Down*
Fonte: One more Agile Blog[1]

O uso das métricas é valido quando há uma correta apresentação das mesmas, disponibilizando-as para a equipe e difundindo os pontos positivos da evolução da equipe. Logo, grandes gráficos expostos e de fácil compreensão são os melhores. Uma tabela extensa cheia de dados desordenados não é visivelmente atrativa e, mesmo que contenha uma densa quantidade de informações, pode se tornar menos informativa que um gráfico de linhas contendo os mesmos dados.

Segundo [1], pode vir a ser perigoso o uso de estatísticas na apresentação de métricas. Isso por que estatísticas são facilmente maleáveis, possibilitando que uma boa amostra seja apresentada como um ponto negativo ou pejorativo para algum membro da equipe. Uma possibilidade de uso de estatísticas é com visão do time todo por iterações ou *releases*. Desta forma não há uma especificação do individuo, dificultando que possam ser apontados atores responsáveis por um possível baixo desempenho do time.

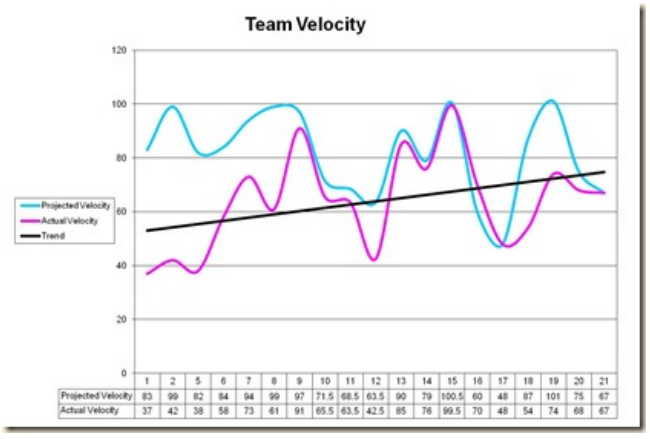

Figura 2. Exemplo de gráfico de *Velocity*
Fonte: One more Agile Blog[1]

Em sistemas Linux, o uso de *scripts* é recorrente e facilitam a automação de ações, deixando transparente para o usuário algumas interferências [2]. Aproveitando o uso de ferramentas de integração continua, é possível gerar diversos relatórios que representam uma evolução detalhada do projeto. O uso combinado das ferramentas com *scripts* de validação e retirada de métricas é bastante comum no desenvolvimento de aplicações nos dias atuais. Ferramentas como o Pylint podem ser utilizadas em curtos intervalos de desenvolvimento, providenciando métricas como cobertura e qualidade de código em números, o que possibilita a produção de gráficos informativos.

[1]Disponível em: <`http://www.onemoreagileblog.com/2009/07/common-agile-metrics.html`>. Acesso em 15 de outubro de 2014

3 Metodologia

[Consiste na maneira de trabalhar o objeto de pesquisa, as ações pelas quais serão alcançados os resultados esperados da pesquisa. Descrever:

- processo de estudo das fontes bibliográficas
- Instrumentos e fontes para coleta de dados
- Processos e métodos para a realização do trabalho (uso de ferramentas, linguagens de especificação, descrição das fases do trabalho de pesquisa, etc)

]

4 Equipe

Alunos

- Rafael Fazzolino Pinto Barbosa
- Thiago Ramires Kairala
- Eduardo Brasil Martins
- Bruno Contessotto Bragança

Professores

- Mr. George Marscicano

Referências Bibliográficas

[1] CRISPIN, L.; GREGORY, J. Agile testing: A practical guide for testers and agile teams. [S.l.]: Pearson Education, 2009.

[2] MITCHELL, M.; OLDHAM, J.; SAMUEL, A. Advanced Linux Programming. [S.l.], 2001.