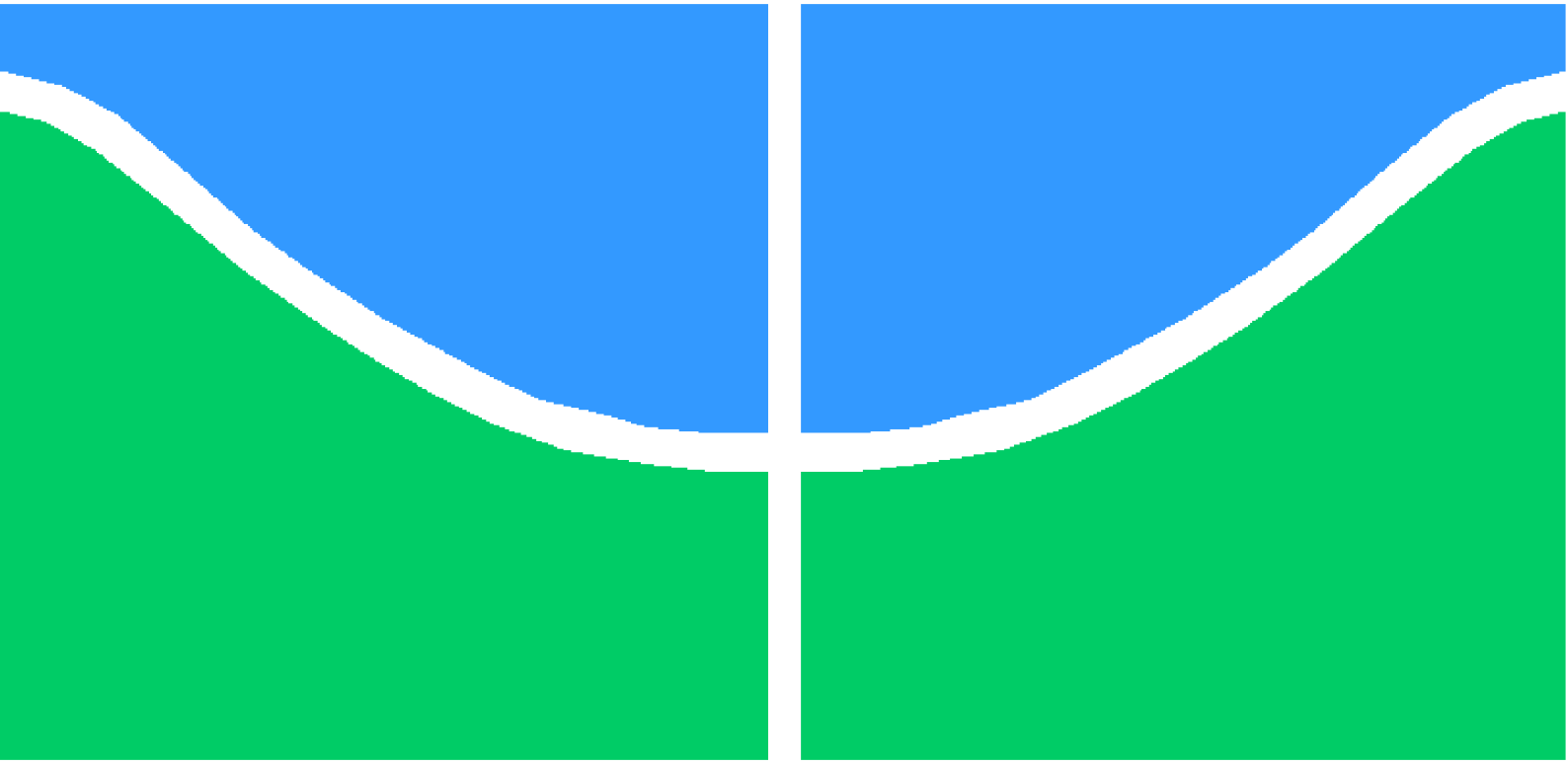

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de Software

Criação de um Dashboard para Monitoramento de Perfis de Qualidade de Software

Autor: Levi Moraes dos Santos

Orientador: Prof. Dr. Maurício Serrano
Coorientadora: Profa. Dra. Milene Serrano

Brasília, DF

2016

Levi Moraes dos Santos

Criação de um Dashboard para Monitoramento de Perfis de Qualidade de Software

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software .

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. Dr. Maurício Serrano

Coorientador: Profa. Dra. Milene Serrano

Brasília, DF

2016

Levi Moraes dos Santos
Criação de um Dashboard para Monitoramento de Perfis de Qualidade de Software / Levi Moraes dos Santos. – Brasília, DF, 2016-
41 p. : il. ; 30 cm.

Orientador: Prof. Dr. Maurício Serrano

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2016.

1. Qualidade. 2. Dashboard. I. Prof. Dr. Maurício Serrano. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Criação de um Dashboard para Monitoramento de Perfis de Qualidade de Software

CDU 02:141:005.6

Levi Moraes dos Santos

Criação de um Dashboard para Monitoramento de Perfis de Qualidade de Software

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software .

Trabalho aprovado. Brasília, DF, 01 de junho de 2013:

Prof. Dr. Maurício Serrano
Orientador

Profa. Dra. Milene Serrano
Coorientadora

---

s
Convidado 1

Convidado 2

Brasília, DF
2016

•*Dedicatória ao Sr 1,2,3 de Oliveira 4*

Agradecimentos

Agradeço inicialmente a mim mesmo, porém agradecerei a mais pessoas quando me formar

Resumo

Palavras-chaves: qualidade. dashboard. monitoramento de métricas. visualização de métricas.

Abstract

This is the english abstract.

Key-words: latex. abntex. text editoration.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

MIT	Massachusetts Institute of Technology
ISO	International Organization for Standardization

Sumário

1 Introdução

Neste primeiro capítulo é apresentado uma visão mais ampla do trabalho e que tem como objetivo introduzir a temática abordada. Este capítulo está dividido em 6 (seis) seções. Na primeira seção é abordado o contexto (Contextualização) em que se encontra este trabalho. Seguido da contextualização apresenta-se a problematica (Problema) na qual se deseja resolver com a solução apresentada. As justificativas (Justificativa) que levaram à realização deste trabalho. Após as justificativas são apresentados os objetivos (Objetivos) que servem como guia para solução proposta. Resultados esperados (Resultados Esperados) que como o próprio nome já sugere apresenta o que é esperado ao fim deste trabalho e por último organização do Trabalho (Organização do Trabalho) que descreverá o conteudo apresentado durante todo o trabalho.

1.1 Contextualização

O conceito de engenharia de software foi proposto inicialmente durante uma conferência na década de 60 em Garmisch na Alemanha. Nesta conferência estavam presentes usuarios, fabricantes e pesquisadores que debatiam sobre os constantes problemas no desenvolvimento de software [Paduelli 2007]. A qualidade na produção de software é uma área muito ampla e que abrange desde qualidade da arquitetura de software até qualidade no processo REFERENCIA. Este trabalho teve como objetivo central apresentar uma solução para visualização de métricas de qualidade dentro de alguns orgãos, tendo como os principais pilares três áreas comuns da engenharia de software, gerência de configuração, integração contínua e análise estática de código. Uma arquitetura próxima a essa vem sendo trabalhada dentro de alguns órgãos públicos do governo federal, entre eles o Tribunal de Contas da União o qual tem mostrado os melhores resultados nesta área. O problema encontrado atualmente está na falta de um acompanhamento na qualidade dos chamados softwares legados, softwares que são produzidos dentro/para o órgão e que passado um tempo ainda estão em atividade.

Dentro da área de Tecnologia as organizações tem encontrado um grande problema quando se trata de softwares legados. O trabalho feito por REFERENCIA prova que os softwares legados são esquecidos pelas organizações quando se fala sob uma perspectiva de manutenção. Contudo o estudo também revela que em grande parte das empresas esses mesmos softwares continuam rodando no ambiente de produção.

1.2 Problema de pesquisa

O principal produto da engenharia de software é o software, contudo o que tem se vivenciado na realidade brasileira de computação é que o software que está sendo entregue é um software precário e de baixa qualidade. Por ser uma palavra abstrata, o conceito de qualidade é bem amplo, porém o termo qualidade normalmente está associado a uma medida relativa, essa qualidade pode ser entendida como "conformidade às especificações". Conceituando dessa forma, a não conformidade às especificação é igual a ausência de qualidade.
Uma das grandes dificuldades nos orgãos públicos está no acompanhamento das manutenções prestadas por terceirizadas. Esse problema se agrava ainda mais quando a empresa contratante não consegue acompanhar ou não tem parametros concretos de indicadores de qualidade. Este trabalho tem como proposta a criação de uma dashboard de monitoramento para softwares legados onde é possível acompanhar de maneira simples e totalmente visual, indicadores de qualidade de código para projetos selecionados.
O grande desafio está em criar uma maneira de visualização de dados de maneira simplificada e objetiva para acompanhamento de software em um orgão público federal. Tomando este problema como base tem-se a seguinte questão de pesquisa:

"Uma vez definido os indicadores das métricas, como criar uma interface de visualização da informação? "

1.3 Justificativa

A motivação deste trabalho se deu como uma extensão de um trabalho de conclusão de concurso elaborado anteriormente em que eram tratados aspectos para monitoramento da qualidade dentro de um orgão público federal. O trabalho abordava aspectos de contratação de software dentro desses orgãos e como acontecia o acompanhamento desses softwares e propunha uma solução com integração contínua e gerência de configuração.
Após a conclusão do trabalho citado algumas lacunas continuaram, essas lacunas estão ligadas à apresentação destes dados para a equipe de gestão com o intuito de facilitar o acompanhamento das métricas.

1.4 Objetivos

1.4.1 Objetivos Gerais

Desenvolver uma solução intuitiva para monitoramento da qualidade de código de software

1.4.2 Objetivos Específicos

Para que seja possivel alcancar o objetivo geral alguns outros objetivos menores precisam ser alcancados para garantir o objetivo geral

- Identificar metricas de codigo ja existentes que mais se adequem as necessidades de um projeto de software
- Propor um ambiente integralizado e automatizado, englobando solucoes de analise estatica de código, integracao continua e versionamento de codigo
- Uma solução que agregue valor à equipe de manutenção

1.5 Resultados Esperados

1.6 Organização do trabalho

2 Referencial teórico

Este capítulo tem como objetivo servir como referencial teórico para todo o documento. As idéias discutidas neste capítulo são

2.1 Qualidade

O principal produto da engenharia de software é o software, contudo o que tem se vivenciado na realidade brasileira de computação é que o software que está sendo entregue é um software precário e de baixa qualidade. Por ser uma palavra abstrata, o conceito de qualidade é bem amplo, porém o termo qualidade normalmente está associado a uma medida relativa, essa qualidade pode ser entendida como "conformidade às especificações". Conceituando dessa forma, a não conformidade às especificação é igual a ausência de qualidade [Paduelli 2007].

A ISO 9126-1 proposta em 2001, também conhecida como Engenharia de Software - Qualidade do Produto, descreve o modelo de qualidade voltado para o produto de software como sendo composto por duas categorias como pode ser visto na Figura 1. A primeira categoria está relacionada a qualidade interna e a qualidade externa do software. A segunda categoria se relaciona com a qualidade de uso do software [NBR ISO/IEC 9126-1 2016]

Sob o aspecto de modelo de qualidade,a ISO 9126 classifica a qualidade interna do produto como sendo o somatório das características do ponto de vista interno do software. Os principais produtos desta categoria são os de cunho intermediário, entre eles: relatórios de analise estática do código fonte, revisão dos documentos produzidos, entre outros.

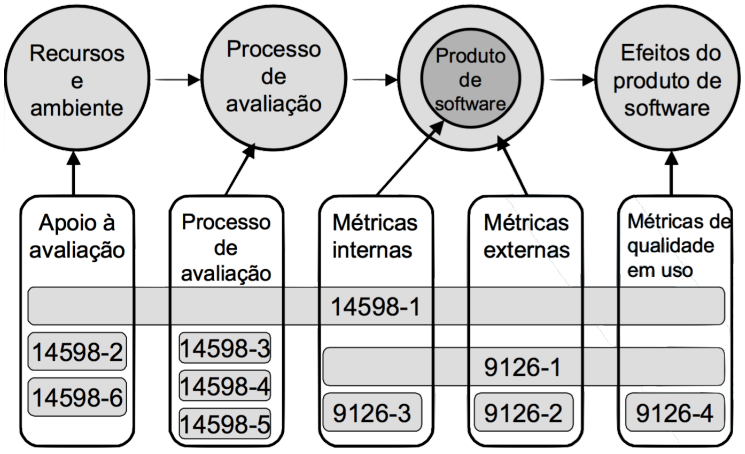

Figura 1 – Relação entre as NBR ISO/IEC 9126 e NBR ISO/IEC 14598 .Fonte: [NBR ISO/IEC 9126-1 2016]

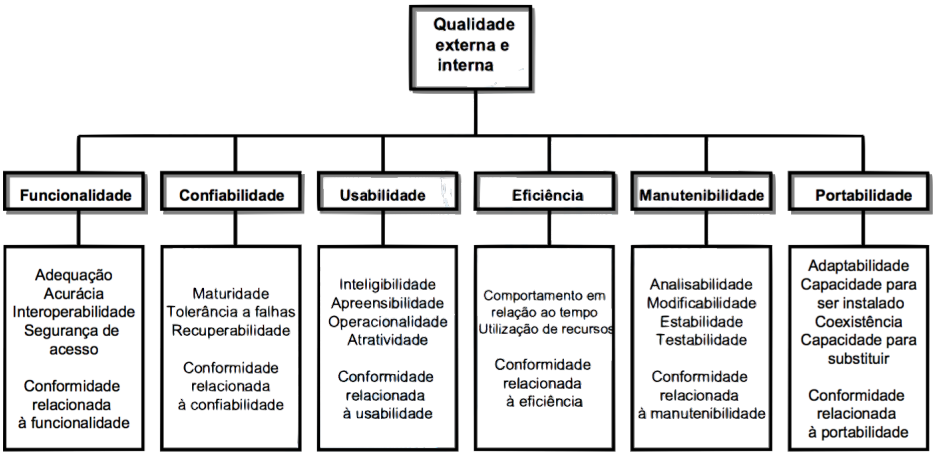

Figura 2 – Modelo de Qualidade para Qualidade Interna e Externa .Fonte: [NBR ISO/IEC 9126-1 2016]

A qualidade externa por sua vez já apresenta o seu foco mais voltado para as relações externas do software, normalmente esta relacionado com a execução do código coletando suas métricas enquanto o software está em funcionamento. A Figura 2 apresenta a divisão proposta pela [NBR ISO/IEC 9126-1 2016] onde são categorizados seis aspectos de qualidade de software e suas subcaracterísticas, essas podem ser medidas por meio de métricas internas e externas.
Segundo a ISO 9126 essas caracteristicas podem ser definidas como:

- **Funcionalidade**: Capacidade do produto de software de prover funções que atendam às necessidades explícitas e implícitas, quando o software estiver sendo utilizado sob condições especificadas.

- **Confiabilidade**:Capacidade do produto de software de manter um nível de desempenho especificado, quando usado em condições especificadas.

- **Usabilidade**: Capacidade do produto de software de ser compreendido, aprendido, operado e atraente ao usuário, quando usado sob condições especificadas.

- **Eficiência**: Capacidade do produto de software de apresentar desempenho apropriado, relativo à quantidade de recursos usados, sob condições especificadas.

- **Manutenibilidade**: Capacidade do produto de software de ser modificado. As modificações podem incluir correções, melhorias ou adaptações do software devido a mudanças no ambiente e nos seus requisitos ou especificações funcionais.

- **Portabilidade**: Capacidade do produto de software de ser transferido de um ambiente para outro.

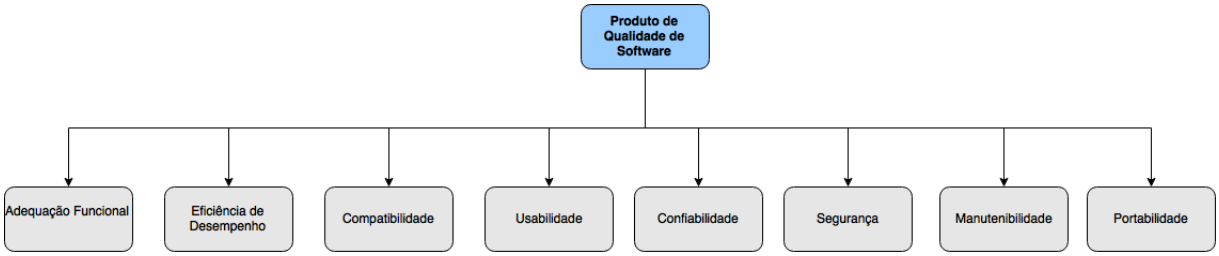

Figura 3 – Produto de Qualidade de Software.Fonte: [ISO 25010]

Em 2011 surgiu um conjunto de normas conhecidos como SQuaRE que traziam um framework aprimorado à atual norma vigente. Este framework tinha como objetivo avaliar o produto de qualidade de software.

2.1.1 Normas SQuaRE

Este conjunto de normas surgiu para substituir a ISO/IEC 9126, Engenharia de Software - Qualidade do Produto. ISO/IEC 25010 mantém as características de qualidade com alguns incrementos.

- O escopo dos modelos de qualidade foram extendidos para incluir sistemas computacionais e a qualidade em uso pelo ponto de vista do sistema
- Segurança foi adicionada como característica, e não uma subcaracterística de funcionalidade.
- Compatibilidade foi adicionada como característica.
- A qualidade interna e externa foram combinadas como modelo de qualidade de produto.

A norma apresenta três guias de qualidade. O primeiro modelo é referente à Qualidade do Produto, o segundo da Qualidade em Uso e o último Qualidade de Dados.O modelo de Qualidade do Produto subdivide um sistema de software em oito categorias como mostra a imagem 3.

Este trabalho tem seu desenvolvimento focado no modelo de Qualidade de Uso que apresenta características externas ao software [ISO 25010]. O foco deste trabalho está em medir indicadores quanto à manutenabilidade do software. Essa característica está diretamente ligada ao processo de Manutenção do Software.

2.1.2 Manutenção de Software

Segundo Sommervile [Sommervile 2011] manutenção de software é o processo de alterar o sistema depois que ele foi publicado. As alterações feitas no software podem ser simples correções de erro, a até mudanças significativamente grandes que corrigem falhas arquiteturais, ou mesmo melhorias para acomodar novos requisitos.
Outra visão sobre manutenção de software é dada por Pressman [Pressman 2010] em que o autor conceitua o termo como sendo a correção de defeitos, adaptação do software para atender uma mudança do ambiente e aperfeiçoar as funcionalidades para que atendam às necessidades dos usuários. Outra característica do processo de manutenção é a sua composição por um conjunto de sub processos, atividades e tarefas que podem ser utilizados durante a fase de manutenção para alterar um produto de software, contanto que seja mantido o seu funcionamento [Calazans e Oliveira 2005].
Para Sommervile existem quatro categorias de manutenção:

- **Manutenção Corretiva**: seu objetivo está em identificar e remover falhas de software
- **Manutenção Adaptativa**: provê modificações no software para alojar mudanças no ambiente externo. Nesta manutenção também está incluso o processo de migração para diferentes plataformas tanto de software quanto de hardware.
- **Manutenção Perfectiva**: está manutenção é feito com o intuito de aperfeiçoar o software, além dos requisitos funcionais originais. Esta expansão dos requisitos traz consigo uma melhoria às funcionalidades até então implementadas ou um ganho de performance do sistema.
- **Manutenção Preventiva**: implementada para permitir que seja mais simples a correção, adaptação ou melhoria do software.

O modelo da figura [**?**] apresenta as atividades propostas por Pfleeger para um processo de manutenção. Na figura percebe-se que o processo de acompanhamento da manutenção ocorre durante todo o processo. As atividades apresentadas no diagrama são:

- **Analise do Impacto da Mudanção de Software**: estima o impacto de uma determinada mudança. Nesta atividade determina-se o grau de mudança e o quanto está mudança impactará no resto do software.
- **Entendimento do Software a ser Alterado**: nesta atividade são analisados os códigos-fonte do software para entender a mudança e a integração do que deve ser alterado. Esta atividade depende muito do grau de manutenabilidade do software, uma vez que quanto mais manutenível mais fácil e rapido se dá o processo de analise do software.

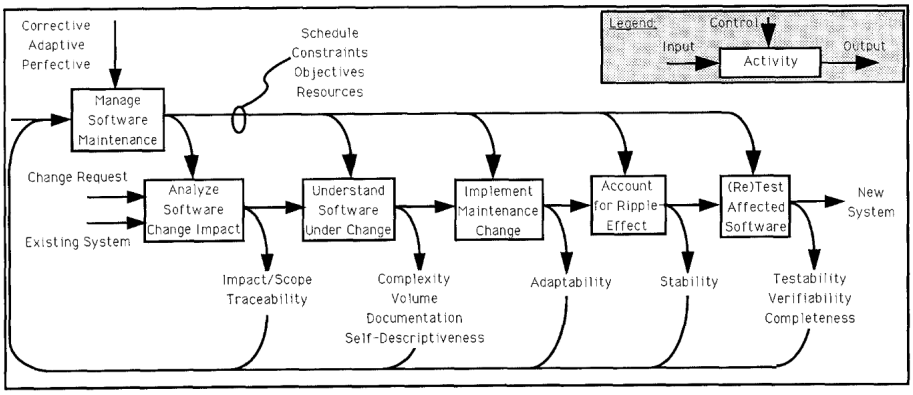

Figura 4 – Diagrama das Atividades de Manutenção de Software.Fonte: [Pfleeger e Bohner 1990]

- **Implementação da Mudança**: incremento ou modificação do software. Está atividade é diretamente relacionada com o grau de adaptação do software, o quanto o software pode ser expandido ou comprimido. Essa característica de adaptabilidade é uma subcaracterística da Manutenabilidade de software apresentada pela norma Square.

- **Mudanças pelo Efeito Cascata**: Analise da propagação das mudanças ao longo do software. Essa atividade está intimamente relacionada com o indicador de coesão e acomplamento do software, este afere o quão amarrado estão as classes e os métodos do software.

- **(Re)Teste do Software**: é a ultima atividade antes da entrega do software alterado. O software é testado novamente sob a perspectiva do novo requisito.

Um estudo realizado por Kustyers e Heemstra [Kusters e Heemstra 2001] mostram a dificuldade de atuais na manutenção de software em seis grandes organizações da Alemanha. Um dos resultados obtidos foi que existe um uma falta muito grande na percepção quanto ao tamanho e o custo das manutenções de software. Os autores relatam que os gastos com manutenção são altos e que das seis empresas apenas uma mantinha registrado os seus processos de manutenção e os usava para fazer um novo planejamento. Normalmente tem se como verdade de que a manutenção de software está unicamente ligada ao conserto de *bugs*, entretanto estudos e *surveys* ao longo dos anos comprovam que mais de 80% do esforço gasto na manutenção é utilizado em ações não corretivas segundo Pigosky [Pigoski 1996]. O autor também afirma que entre 40% a 60% do esforço de manutenção está em entender o software que será modificado.

3 Suporte Tecnológico

O objetivo desta seção é explicitar todo o aparato tecnológico a nível de software que foi utilizado durante o desenvolvimento deste trabalho. Esta seção está dividida em *Ferramentas para programação*.

3.1 Ferramentas para Programação

Neste tópico, serão apresentadas ferramentas e tecnologias voltadas ao contexto da Engenharia de Software que são utilizadas durante este trabalho, como, por exemplo, ferramentas para gerência de configuração e versionamento dos artefatos gerados.

3.1.1 GIT

A ferramenta GIT[1] foi desenvolvida por Linus Torvalds durante a criação do Kernel Linux, pois Linus percebeu que existia a necessidade de criar uma ferramenta *open-source* que fizesse o controle de versão [BENTO et al. 2013].
O motivo para escolha da ferramenta se deve ao fato de que o Git contém o suporte para desenvolvimento linear o que garante um paralelismo de diversas áreas do desenvolvimento. Outro diferencial do Git está nos *snapshots* dos objetos que são armazenados, isso significa que o Git não rearmazena arquivos que não foram alterados [MARTINHO e MUNIZ 2013].

3.1.2 Github

O Github[2] é um repositório *online* que fornece a criação de projetos públicos gratuitos. A ferramenta também provê um sistema de gestão para acompanhamento do desenvolvimento envolvendo um sistema de *logs*, gráficos de visualização e uma *Wiki* integrada a cada projeto [MARTINHO e MUNIZ 2013].

O principal motivo pela escolha do Github é que deseja-se disponibilizar a solução futuramente para consulta e aprimoramento de pessoas interessadas.

3.1.3 Bonita

A ferramenta Bonita[3] foi escolhida graças a sua facilidade de utilização e portabilidade para o sistema operacional Mac OS X e o fato de ser gratuita. Esta ferramenta

[1] https://git-scm.com/
[2] https://github.com
[3] http://www.bonitasoft.com/

auxilia na modelagem de processos.

3.1.4 Mac OS X

O sistema operacional Mac OS foi baseado no kernel Unix e fabricado e desenvolvido pela empresa Apple Inc. Utilizou-se a versão 10.11 do sistema também conhecida como "*El Capitan*".

3.1.5 LaTeX

O LaTeX[4] foi desenvolvido na década de 80 cujo objetivo era simplificar a diagramação de textos científicos e matemáticos, onde atualmente dispõe de uma grande quantidade de macros para bibliografia, referencias, gráficos entre outros.

3.1.6 Sublime Text 3

O Sublime Text 3[5] é um editor de texto bastante utilizado por programadores, por possuir apoio para diversas linguagens de programação, incluindo textos em LaTeX.

3.1.7 Zotero

Zotero [6] é um software para gerenciamento de referências bibliográficas. Ele possui integração com o *browser*, sincronização online e criação de bibliografias estilizadas.

[4] https://www.latex-project.org/
[5] https://www.sublimetext.com/3
[6] https://www.zotero.org

4 Proposta

5 Metodologia

5.1 Metodologia

A partir da Metodologia

6 Resultados Parciais

7 Considerações finais

Considero nada a ninguem

Referências

BENTO, G. V. et al. Análise Comparativa de Sistemas de Controle de Versões com foco em Versionamento de Banco de Dados Oracle. *e-RAC*, v. 3, n. 1, 2013. Disponível em: <http://www.computacao.unitri.edu.br/erac/index.php/e-rac/article/view/122>. Citado na página 31.

CALAZANS, A. T. S.; OLIVEIRA, M. A. L. Avaliação de Estimativa de Tamanho para projetos de Manutenção de Software. In: *Proc. of Argentine Symposium on Software Engineering*. [S.l.: s.n.], 2005. Citado na página 28.

ISO 25010. *Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models*. 2011. ed. [S.l.]. Citado 2 vezes nas páginas 13 y 27.

KUSTERS, R. J.; HEEMSTRA, F. J. Software maintenance: an approach towards control. In: *Software Maintenance, 2001. Proceedings. IEEE International Conference on*. [S.l.: s.n.], 2001. p. 667–670. ISSN 1063-6773. Citado na página 29.

MARTINHO, M.; MUNIZ, E. GIT SCM: Sistema de Controle de Versionamento Distribuído. *GIT SCM: Sistema de Controle de Versionamento Distribuído*, 2013. Disponível em: <http://www.tjam.jus.br/index.php?option=com_content&view=article&id=4032%3Agit-scm-sistema-de-controle-de-versionamento-distribuido&catid=344%3Asds-artigos-cientificos-e-tecnologicos&Itemid=455&showall=1>. Citado na página 31.

NBR ISO/IEC 9126-1. [S.l.], 2016. v. 5, n. 7, 088–108 p. Disponível em: <http://periodicos.udesc.br/index.php/reavi/article/view/2316419005072016088>. Citado 3 vezes nas páginas 13, 25 y 26.

PADUELLI, M. M. *Manutenção de Software: problemas típicos e diretrizes para uma disciplina específica*. Dissertação (Mestrado), 2007. Citado 2 vezes nas páginas 21 y 25.

PFLEEGER, S. L.; BOHNER, S. A. A framework for software maintenance metrics. In: *Software Maintenance, 1990, Proceedings., Conference on*. [S.l.]: IEEE, 1990. p. 320–327. Citado 2 vezes nas páginas 13 y 29.

PIGOSKI, T. M. Pratical software maintenance: Best practices for managing your software investment. *Wiley Computer Publishing*, 1996. Citado na página 29.

PRESSMAN, R. S. *Software Engineering - A Practioner's Approach*. 7. ed. [S.l.]: Higher Education, 2010. Citado na página 28.

SOMMERVILE, I. *Software Engineering*. 9. ed. [S.l.]: Pearson, 2011. Citado na página 28.