Faculdade de Tecnologia de Garça "Deputado Julio Julinho Marcondes de Moura"

ANDRESSA ROBERTA SIQUEIRA DOS SANTOS

RELATÓRIO CIENTÍFICO FINAL

GARÇA
2016

Faculdade de Tecnologia de Garça "Deputado Julio Julinho Marcondes de Moura"

ANDRESSA ROBERTA SIQUEIRA DOS SANTOS

RELATÓRIO CIENTÍFICO FINAL

Relatório Final apresentado a Comissão Permanente de Pesquisa da Faculdade de Tecnologia "Dep. Júlio Julinho Marcondes de Moura" – FATEC, como requisito para emissão do parecer técnico-científico do projeto de Iniciação Científica.

Orientador: João Baptista Cardia Neto

GARÇA
2016

RESUMO

A medida que as indústrias relacionadas a tecnologia de informação expandem sua presença na economia, a atividade de software ganha cada vez mais relevância. Esse processo de expansão e amadurecimento do mercado fez com que as organizações fossem pressionadas por seus concorrentes, de forma a serem cada vez mais capazes de integrar e acelerar seus processos de desenvolvimento de software. A obtenção de qualidade nos projetos torna-se um importante diferencial competitivo. Assim, foi realizado um estudo de caso, verificando o processo de desenvolvimento de um software de gestão de taxistas e mototaxistas. As métricas estabelecidas permitiram verificar como os dados gerados durante o próprio processo - se corretamente verificados – permitem identificar problemas e inconsistências de forma precoce, permitindo que os gestores realizem os ajustes necessários para melhorar a qualidade do produto entregue ao cliente.

***Palavras-chave**: Qualidade de Software. Melhoria Contínua. Engenharia de Software*

SUMÁRIO

INTRODUÇÃO

A preocupação com a qualidade do software teve seu crescimento na década de 90 quando as empresas notaram que bilhões de dólares eram desperdiçados quando seus produtos não apresentavam as funcionalidades e características prometidas. Dessa forma, foram desenvolvidos processos e práticas que garantissem não só a qualidade mas diminuíssem custos e prazos para a colocação dos produtos no mercado. É sabido que a qualidade de um produto é influenciada pela qualidade do processo de produção. Inúmeros produtos são lançados diariamente, com diversas funcionalidades e seu funcionamento é um dos diferenciais durante a escolha dos consumidores.

Pesquisa sobre o mercado brasileiro de software da Associação Brasileira de Empresas de Software apontou que, com relação aos indicadores de Mercado e Evolução de 2004 – 2014, o mercado brasileiro teve um crescimento de 12,8% em relação a 2013, representando 2,9% do mercado mundial (MERCADO, 2015).

Assim sendo, o objetivo da pesquisa é verificar o ciclo de desenvolvimento de um software e as ferramentas utilizadas para gerenciamento desse ciclo, demonstrando como os dados disponibilizados pelos sistemas permitem a identificação dos principais gargalos e rotinas que apresentaram mais problemas, permitindo ao gestor do projeto a definição de estratégias de melhoria do processo. Assim, os dados finais servirão de apoio as estratégias de negócio.

Esta pesquisa apresenta também os resultados do estudo de caso que procurou investigar um software existente há pouco menos de cinco anos no mercado e que tem como finalidade auxiliar a regulamentação e administração da prestação de serviço de moto táxi, apresentando relatórios gerenciais, registros de ocorrências, controle de vagas dentre outras funcionalidades, sendo gerenciado pelo JIRA, sistema que dispõe de ferramentas essenciais para o gerenciamento das tarefas realizadas pela equipe de desenvolvimento.

As informações foram obtidas do JIRA, da documentação do projeto, análise de requisitos e manual. Como fruto deste trabalho foi constatado que os dados de Esforço e Complexidade - que juntos proporcionaram uma métrica para análise das rotinas tanto de forma individual quanto coletiva - contribuem no estabelecimento de uma visão geral do software. Confirmou-se que é possível identificar deficiências no processo de desenvolvimento com base nesses dados.

JUSTIFICATIVA

A pesquisa apresenta relevância teórica e prática uma vez que possibilita o exercício de análise da documentação do software e dos dados disponibilizados durante as coletas, além de produzir base para pesquisas futuras aprofundando a discussão sobre boas práticas de gerenciamento de projeto e a evolução dos processos e produtos no ciclo de desenvolvimento.

OBJETIVOS

O principal objetivo da pesquisa é demonstrar como a melhoria da qualidade do processo de desenvolvimento como um todo pode ser realizada com os dados disponíveis pela própria ferramenta utilizada para administrá-lo.

A fim de que o objetivo específico seja alcançado, buscou-se atingir os seguintes objetivos intermediários:
- Verificar como se dá o gerenciamento do projeto;
- Verificar a relação entre a qualidade de processo e produto;
- Compreender o contexto da emergência das novas metodologias de desenvolvimento.

REVISÃO BIBLIOGRÁFICA

A evolução de uma sociedade industrial para uma sociedade da informação culminou na emergência da economia do conhecimento, com destaque para a indústria de software, protagonista de um conjunto de mudanças tecnológicas. Segundo Meira e Araújo (2004) essa indústria pode ser desenvolvida em qualquer região que possua os requisitos básicos de um sistema de inovação e tem como característica em geral a predominância de pequenas empresas que segundo a ABES (Associação Brasileira de Empresas de Software) lideram cerca de 45,62% e 49,02% do mercado se considerarmos micro e pequenos negócios.[1]

A medida que as indústrias relacionadas a tecnologia de informação expandem sua presença na economia, a atividade de software ganha cada vez mais relevância. Dados de 2003 indicavam que a indústria de software no Brasil ultrapassara 7 bilhões de dólares em faturamento. Já em 2014 o faturamento do mercado de software atingiu a marca de U$11,12 bilhões.[2]

O processo de expansão e amadurecimento do mercado fez com que as organizações fossem pressionadas por seus concorrentes. Assim, deveriam ser capazes de integrar e acelerar seus processos de desenvolvimento de software, e a obtenção de qualidade nos projetos torna-se um importante diferencial competitivo. Dessa forma, principalmente a partir da década de 80 surgiram diversos modelos de qualidade e maturidade com o objetivo de assegurar os processos relativos aos produtos de software.

Dentre as principais abordagens resultantes da maior preocupação com a melhoria do processo e modelagem da produção de software com qualidade, pode-se destacar o *Capability Maturity Model Integration* (CMMI) e as normas ISO/IEC 15504, ISO 9000 e ISO/IEC 12207 que asseguram que a melhoria do produto é decorrência da melhoria do processo de software. Diferente do modelo do qual derivou [nota no rodapé][3], o CMMI integra aspectos de definição do produto e processos, que além de organizar uma estrutura baseada em níveis procura formar alicerces com vistas a melhoria contínua de processos. A partir do momento em que um conjunto de objetivos do processo é satisfeito, diz-se que um componente do processo foi estabilizado.

As normas ISO por sua vez definem um modelo para avaliação de processo, que pode ser utilizado como referência para a melhoria de processo de software (ISO/IEC 15504); definem processos do ciclo de vida do software (ISO/IEC 12207) e por fim a

---

[1] Dados disponibilizados no mês de maio pela entidade. Disponível em: < http://www.abessoftware.com.br/noticias/investimentos-em-ti-somam-us-60-bilhoes-em-2014>

[2] Informações disponíveis em < http://corporate.canaltech.com.br/noticia/mercado/brasil-e-o-7o-no-mundo-em-investimentos-em-ti-com-us-60-bilhoes-em-2014-40965/>

[3] A diversidade de modelos no CMM gerou problemas tais como conceitos recebendo nomes diferentes em cada modelo. Outras limitações referem-se a altos custos de treinamentos em organizações que optassem pela utilização de mais de um modelo

norma ISO 9000 buscou definir o conceito de qualidade como "a totalidade das características de uma entidade que lhe confere a capacidade de satisfazer às necessidades explícitas e implícitas" (COLOMBO; GUERRA, 2009).

São as atividades de garantia da qualidade as responsáveis por definir a estrutura para se atingir a qualidade de software dentro padrões e procedimentos adequados. O gerenciamento de qualidade por sua vez assegura-se que o nível de qualidade seja atendido. Nesse contexto, há uma relação entre a qualidade de processos e a qualidade de produto que não deve ser desconsiderada uma vez que ambas em conjunto influenciam o nível de qualidade do software. Assim enquanto padrões de processo definem características relativas aos processos, como definições de especificação, validações e descrição de documentos, controle de mudanças, de registro de teste ou de liberação de versão os padrões de produtos definem características a serem seguidas quanto ao formato de plano de projeto, estilo de programação ou estrutura de documento de requisitos (SOMMERVILLE, 2003).

Ainda que vários modelos e padrões tenham sido criados e aprimorados ao longo dos anos, as pressões no mercado por inovação, flexibilidade e produtividade resultaram em novas abordagens no processo de desenvolvimento. A partir do momento em que as datas de entrega do software tornaram-se cada vez mais curtas e os requisitos estavam cada vez mais dinâmicos e passíveis de alteração, o modelo clássico de desenvolvimento já não comportava as mudanças vivenciadas.

Há 15 anos foi lançado o Manifesto Ágil e estabelecida a Aliança Ágil. Especialistas reuniram-se com o objetivo de padronizar seus processos, alicerçados em quatro princípios que buscavam valorizar: Indivíduos e interação entre eles mais que processos e ferramentas; Software em funcionamento mais que documentação abrangente; Colaboração com o cliente mais que negociação de contratos e; Responder a mudanças mais que seguir um plano (BECK, 2001). Em contraste com a metodologia tradicional cujo planejamento focava no projeto como um todo e preocupava-se com o cumprimento rígido de processos, os métodos ágeis acreditavam na entrega do produto ao fim de cada iteração, que por sua vez seriam curtas e com entregas rápidas e objetivas.

Existem diversos métodos ágeis, tais como o *Pragmatic Programming, Feature Driven Development, Adaptive Software Development, Dynamic Systems Development Method, Crystal, Test Driven Development, Scrum e Extreme Programming (XP).* Todos sendo guiados pelos doze princípios do Manifesto Ágil. São eles:

- Nossa maior prioridade é satisfazer o cliente, através da entrega adiantada e contínua de software de valor.
- Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adéquam a mudanças, para que o cliente possa tirar vantagens competitivas.
- Entregar software funcionando com frequência, na escala de semanas até meses, com preferência aos períodos mais curtos.
- Pessoas relacionadas à negócios e desenvolvedores devem trabalhar em conjunto e diariamente, durante todo o curso do projeto.
- Construir projetos ao redor de indivíduos motivados. Dando a eles o ambiente e suporte necessário, e confiar que farão seu trabalho.
- O Método mais eficiente e eficaz de transmitir informações para, e por dentro de um time de desenvolvimento, é através de uma conversa cara a cara.
- Software funcional é a medida primária de progresso.

- Processos ágeis promovem um ambiente sustentável. Os patrocinadores, desenvolvedores e usuários, devem ser capazes de manter indefinidamente, passos constantes.
- Contínua atenção à excelência técnica e bom design, aumenta a agilidade.
- Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito.
- As melhores arquiteturas, requisitos e designs emergem de times auto-organizáveis.
- Em intervalos regulares, o time reflete em como ficar mais efetivo, então, se ajustam e otimizam seu comportamento de acordo. (BECK, 2001)

Visto que o objeto deste estudo de caso é gerenciado segundo o método Scrum, atenta-se especificamente a suas definições.

A METODOLOGIA ÁGIL SCRUM

A primeira referência a Scrum foi feita em 1986 como uma analogia a reunião que ocorre entre jogadores de Rugby ao iniciar um lance. As práticas da metodologia podem ser aplicadas em qualquer contexto pois se adaptam a diferentes cenários. A cada iteração a equipe se divide segundo suas habilidades para entregar o melhor software possível, assim, as equipes auto organizadas acabam realizando a entrega dos softwares de forma incremental.

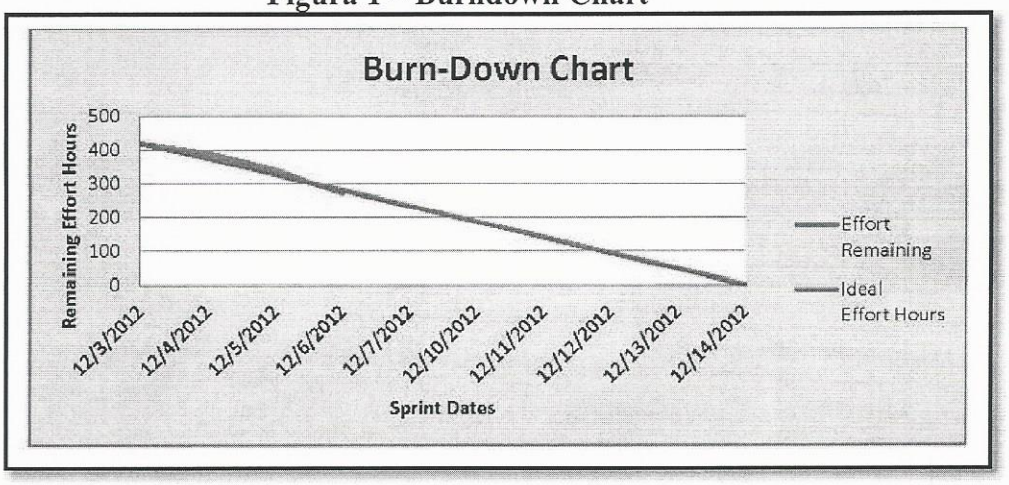
O projeto geralmente é iniciado após a uma visão do produto, que é transformada em requisitos funcionais e não funcionais pelo *Product Owner* (PO). Essa lista, denominada *Product Backlog* segue as prioridades definidas pelo PO. Os itens de maior prioridade serão divididos em tarefas, cuja entrega é determinada pelo Scrum Team. Durante o desenvolvimento, um dos membros da equipe é escolhido com Scrum Master para que seja responsável por remover os impedimentos que surjam durante a realização do trabalho pela equipe.

Os ciclos existentes no processo são denominados Sprints, que tem seu planejamento realizado através de reuniões diárias, reuniões de revisão e de retrospectiva.

> o coração do Scrum é a Sprint, que é uma iteração de um mês ou menos, de duração consistente com o esforço de desenvolvimento. Todas as Sprint utilizam o mesmo modelo de Scrum e todas as Sprint têm como resultado um incremento do produto final que é potencialmente entregável. Cada Sprint começa imediatamente após a anterior (SCHWABER; SUTHERLAND, 2013, p. 7).

O último artefato do Scrum é o *burndown*, que representa graficamente o trabalho realizado e o restante. Mostra-se útil no gerenciamento do projeto ao permitir visualizar quando todo o trabalho será completo (CARVALHO; MELLO, 2012), conforme exemplo a seguir:

Figura 1 – Burndown Chart



Fonte: Scrum Alliance, 2013

METODOLOGIA

Esta seção busca introduzir o tema da pesquisa, apresentando a formulação do problema, resultados, hipóteses e delimitações.

PROBLEMA

Após o despertar da preocupação com a melhoria da qualidade dos sistemas dependentes de software e o estabelecimento de princípios e práticas com o propósito de ajudar as organizações a melhorar sua capacidade de desenvolver seus produtos de forma correta e padronizada, reduzindo riscos e vulnerabilidades, houve também a necessidade de estabelecer métricas para avaliar o grau de melhoria desse processo. Essas métricas seriam utilizadas tanto como apoio nas decisões para gerenciamento, quanto de suporte para análises que definiriam produtividade e qualidade.

Dentro do processo de desenvolvimento de um software existem diversas etapas, cada qual com seu nível de importância. E cada uma delas, passível de melhoria, uma vez que são identificados seus possíveis problemas. Como definimos o que, dentro do processo, deve ser otimizado? Como estabelecemos métodos para medir a eficácia e eficiência de cada processo?

Antes de fazer qualquer tipo de avaliação ou estabelecer métricas, foi realizada uma análise do software em questão e a forma como ele é gerenciado, uma vez que não é possível atingir o objetivo da proposta inicial, qual seja, a melhoria da qualidade do processo de desenvolvimento como um todo, sem antes compreender o software a ser medido e aprimorado e a ferramenta utilizada para administrá-lo, qual seja, o JIRA, que dispõe de inúmeros campos personalizáveis e que são abastecidos pelos gestores do projeto e demais stakeholders.

HIPÓTESE

Foram testadas duas hipóteses: 1) A verificação dos dados disponibilizados pelo software quando transformados em informação são responsáveis pela melhoria do processo. 2) Que a melhoria no processo de desenvolvimento, com a utilização correta das ferramentas se reflete diretamente no aumento da qualidade do software;

Os principais dados que compõe o foco da pesquisa fornecem informações relativas ao esforço e complexidade. Uma terceira hipótese é que, poderemos identificar de forma mais precisas quais as rotinas que, apesar de demandarem mais esforço e serem mais complexas são também aquelas que mais apresentam problemas.

DELIMITAÇÃO DO ESTUDO

A pesquisa pretendeu abordar o processo de desenvolvimento de um software de determinada empresa, cujas especificidades da estrutura organizacional devem ser consideradas. Foram realizadas entrevistas com os gestores do projeto e diretores da empresa de forma a identificar as características do processo que apesar de seguir as boas práticas da engenharia de software ainda apresentava problemas.

A pesquisa não se propõe a mudar a estrutura organizacional nem o processo de desenvolvimento do software uma vez que esta pesquisa caracteriza-se como explicativa e descritiva. Serão realizadas apenas recomendações para os pontos passíveis de melhoria.

TIPO DE PESQUISA

Visto que o objetivo era compreender o funcionamento de determinado sistema sem realizar generalização para outros casos, optou-se por fazer um estudo de caso, que segundo Gil (2002), consiste no estudo profundo e exaustivo de um ou poucos objetos de maneira que permita seu amplo e detalhado conhecimento. Neste tipo de pesquisa, procuramos descrever a estrutura, funcionamento e características de dado objeto de modo a ter um quadro completo sobre o caso.

Para que isso fosse possível, realizou-se então, com base nos procedimentos técnicos utilizados as pesquisas:

- Bibliográfica: obras de diferentes gêneros na área de engenharia de software foram consultadas, como livros, teses e revistas especializadas.
- Documental: os diagramas do sistema também foram consultados de forma a compreender seu funcionamento.

E, com base nos objetivos, realizou-se:

- Descritiva: visando descobrir a existência de associação entre diversas variáveis do sistema (tempo de desenvolvimento, esforço, complexidade), suas características foram descritas após observar seu funcionamento.
- Explicativa: de modo a identificar os fatores que determinaram o fenômeno ocorrido, qual seja, os problemas encontrados no sistema em questão.

UNIVERSO E AMOSTRA

Devido a facilidade de acesso aos dados do sistema, o universo desta pesquisa é o sistema de gerenciamento de taxi e mototaxi de uma microempresa.

COLETA DE DADOS

Os dados foram obtidos mediante diversos procedimentos de forma a garantir a qualidade dos resultados obtidos.

Ainda que tenham sido realizadas observações do funcionamento do sistema, análise de documentos e entrevistas com os gestores, os principais dados foram coletados das issues lançadas no JIRA, responsável pelo gerenciamento do software estudado. As issues por sua vez são classificadas por tipos, podendo ser uma *new feature, bug, improvement* ou *task.*

TRATAMENTO DOS DADOS

Os dados coletados pelas pesquisas foram tratados e ordenados de modo que se possam identificar aspectos relevantes para o estudo.

Os dados levantados foram tratados quantitativamente. A seleção buscou trabalhar com amostras estratificadas, que se caracterizam pela seleção de uma amostra de cada subgrupo da população considerada. Os subgrupos considerados são os tipos utilizados para classificação das issues.

LIMITAÇÕES DO MÉTODO

Considerando o caráter unitário do objeto estudado, o tipo de pesquisa apresenta limitações quanto a generalização dos resultados obtidos.

Além disso, diferente dos outros tipos de pesquisa, no estudo de caso não são definidos procedimentos metodológicos rígidos, o que pode resultar no comprometimento da qualidade dos resultados.

ESTUDO DE CASO

A EMPRESA

O software estudado pertence a uma microempresa situada na cidade de Marília cujo ramo de atividade é o desenvolvimento de softwares para gestão de trânsito.

SISTEMA DE GESTÃO DE TAXI E MOTOTAXI

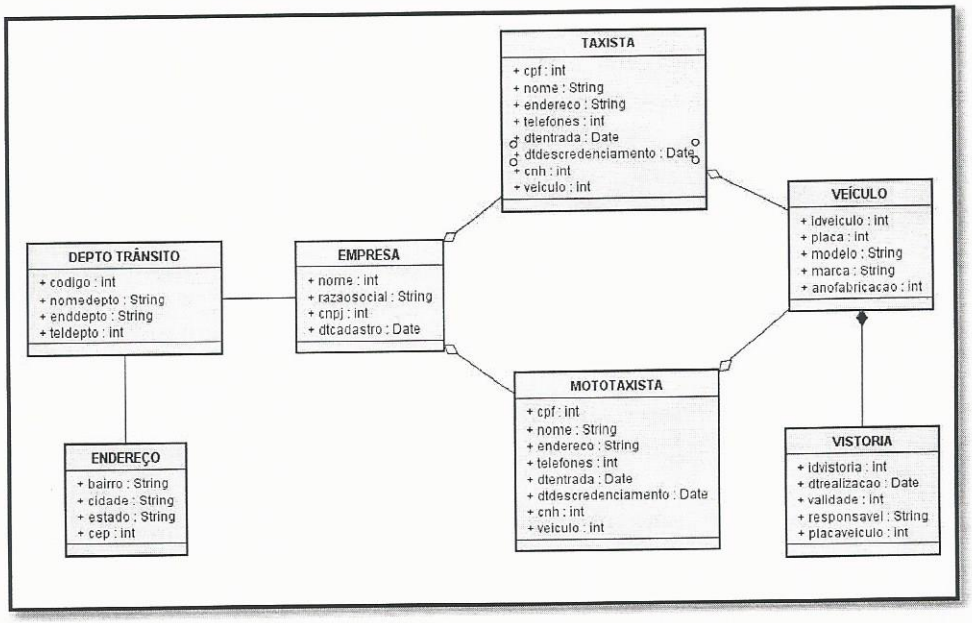
O software em questão foi desenvolvido em plataforma Desk com a função de auxiliar os departamentos de trânsito na gestão e administração dos taxistas e mototaxistas credenciados no município.

Se inicialmente o sistema basicamente contava com cadastros, ao longo dos anos foi recebendo diversas atualizações de forma a atender as novas demandas e superar as expectativas dos clientes. Assim, atualmente, dentre suas principais funcionalidades estão:

- Manutenção dos dados da empresa (CNPJ, nome, razão social, telefone);
- Manutenção dos taxistas e mototaxistas credenciados;
- Manutenção dos dados dos veículos (placa, modelo, marca, ano de fabricação);
- Manutenção e acompanhamento das vistorias dos veículos;
- Manutenção das vagas e pontos dos taxistas e mototaxistas;
- Emissão da credencial dos taxistas e mototaxistas;
- Fornecer relatórios operacionais e gerenciais, servindo de apoio a tomada de decisão.

A estrutura do projeto fica mais clara ao visualizar o diagrama de classes abaixo:

Figura 2 – Diagrama de classes do projeto



Fonte: Elaborado pela autora

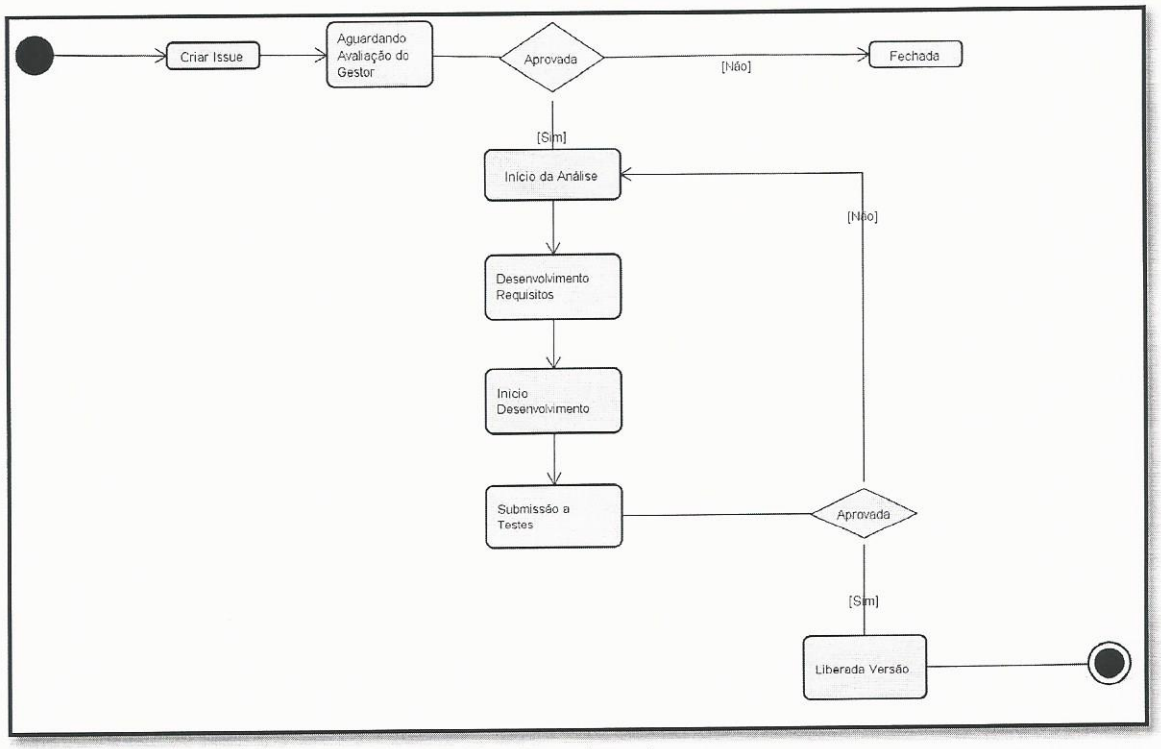
JIRA – GERENCIANDO O PROJETO

A administração do sistema é feita pelo JIRA, ferramenta desenvolvida pela Atlassian em 2002, que além de funcionar como bug tracking, também dispõe de ferramentas essenciais para o gerenciamento das tarefas realizadas pela equipe de desenvolvimento, possibilitando a customização de vários campos e permitindo controlar desde o fluxo de trabalho até as tarefas realizadas, notificações, status, nível de

criticidade e usuários (FISHER & LUDWINGSEN, 2013). Todas as modificações sofridas pelo software exigem que seja lançada uma issue, que orienta todo o processo de desenvolvimento.

As issues são utilizadas de diferentes formas, e podem representar uma requisição, uma tarefa e até mesmo um bug ou correção a ser realizada. Para visualização do fluxo do processo, a ferramenta se utiliza de uma representação gráfica, que permite a melhor visualização do sistema sob diferentes perspectivas e que apesar de exibir uma visão parcial dos elementos, permite que seu acompanhamento seja feito de forma clara e concisa conforme workflow apresentado a seguir:

Figura 3 – Workflow do Processo de Solicitação de Desenvolvimento



Fonte: Elaborado pela autora

O Workflow apresentado demonstra o fluxo de uma solicitação de desenvolvimento de alguma funcionalidade para o software e permite visualizar a estrutura do processo através de uma visão dinâmica do sistema. Uma vez lançada uma solicitação (issue), ela é analisada e, caso o gestor do projeto a julgue pertinente, passa para a fase de análise, desenvolvimento dos requisitos e o processo de desenvolvimento em si. Ao ser submetida aos testes, se reprovada, será novamente analisada. Retorna então para o desenvolvimento e para a fase de testes. Quando liberada, passa a aguardar

até que possa seja liberada e então, uma nova versão do software é gerada, permitindo que a issue seja fechada.

ANÁLISE E DISCUSSÃO DOS RESULTADOS OBTIDOS

Foram analisadas 101 issues disponíveis no Projeto. Essas issues foram classificadas conforme seu tipo. As issues do tipo *Bug* representam problemas em alguma funcionalidade do sistema, as do tipo *Improvement* representam alguma melhoria identificada tanto pelos gestores quanto pelos clientes; *New Features* são solicitações de desenvolvimento de novas rotinas e por fim, *Task* são tarefas relacionadas a issues já lançadas.

Tabela 1 – Quantidade de issues por Tipo

Tipo	Quantidade
Bug	45
Improvement	29
New Feature	25
Task	2

Fonte: Elaborado pela autora

Analisando os dados da tabela acima, podemos afirmar que hoje a correção de erros demanda quase o mesmo esforço que o desenvolvimento de novas rotinas uma vez que 29% das issues lançadas referem-se a melhorias (alterações, inclusões, conversões), 25% são novas funcionalidades, 1% são tarefas relativas ao desenvolvimento do software e 45% envolvem a correção de erros.

Em geral, ao iniciar um novo projeto, as empresas geralmente requerem uma estimativa quanto ao tamanho do software a ser desenvolvido uma vez que interfere no planejamento, negociação de prazos, custos e recursos com o cliente. No entanto, a partir do momento em que o projeto já se encontra consolidado e com certo nível de maturidade, deve-se verificar a melhor forma de acompanhar o esforço do projeto em curso.

Dentro desse contexto, as issues contêm ainda informações sobre esforço e complexidade. A complexidade corresponde as ações requeridas por cada tipo de usuário quanto aos passos indispensáveis para o cumprimento de cada tarefa. Apesar de fazer alusão, não se pode dizer que é utilizada a técnica Pontos por Casos de Uso, uma vez que o objetivo dessa métrica não é estimar o tamanho do sistema e sim pensar no desenvolvimento de uma rotina individualmente. O conceito de complexidade foi apropriado para classificar a complexidade da rotina a ser desenvolvida.

A complexidade é dividida em três níveis: grande, média e pequena. Para o estabelecimento das métricas, atribuímos os pesos 3, 2 e 1 para cada uma respectivamente. Como tal projeto é avaliado por diferentes gestores - não ha como garantir que foram medidas a mesma coisa se os critérios utilizados forem diversificados, assim, para a obtenção de estimativas confiáveis exigiu-se uma padronização da atribuição desse dado com a criação de tópicos gerais para enquadrar as rotinas e foi realizado extenso trabalho de calibração.

Tabela 2 – Quantidade de issues por Complexidade

Complexidade	Quantidade
Grande	17
Média	35
Pequena	49

Fonte: Elaborado pela autora

A maioria das issues lançadas tem baixa complexidade. Assim, 47% das issues são de complexidade pequena, 36% são de média complexidade e 17% são de grande complexidade.

Outro dado importante é o esforço no desenvolvimento da issue. Ele também é utilizado e informado pelo gestor do projeto após a análise da solicitação. Além disso, pode variar de 1 (um) a 5 (cinco).

Tabela 3 – Quantidade de issues por Esforço

Esforço	Quantidade
1	21
2	19
3	24
4	17
5	20

Fonte: Elaborado pela autora

Interessante observar que há certo balanceamento quanto ao esforço no desenvolvimento. 21% das issues do projeto são de esforço 1, 19% são de esforço 2, 24% representam esforço 3, 17% são de esforço 4 e 20% correspondem ao esforço 5.

MEDINDO O PROCESSO

Para determinar o tempo necessário para o desenvolvimento e até mesmo o custo do projeto, suas atividades devem ter suas durações estimadas. Essas estimativas são realizadas com base em métricas, que podem ser de duração, tamanho, produtividade e esforço.

Sommerville (2011) define métrica de software como uma característica de um sistema, documentação ou processo de desenvolvimento que pode ser medido de forma objetiva. Por sua vez, Pressman acredita que a medição de atributos específicos do processo é a única maneira racional de melhorar qualquer processo, uma vez que são desenvolvidas uma série de métricas significativas com base nesses atributos. Ambos os autores trabalham com o conceito de métrica de processo e produto. Para Sommerville, medir o processo não permite que se determine a melhoria na qualidade do produto:

> Process measurements can be used to assess whether or not the efficiency of a process has been improved. For example, the effort and time devoted to testing can be monitored. Effective improvements to

> the testing process should reduce the effort and/or testing time. However, process measurements on their own cannot be used to determine if product quality has improved. (SOMMERVILE, 2011, p. 771)

Roger Pressman (2011), ao trabalhar com as definições de métrica de processo e produto deixa claro que a métrica de processo é utilizada para fins estratégicos e coletada por longos períodos com o objetivo de proporcionar uma série de indicadores que levam a melhoria do processo a longo prazo. As métricas de projeto, por sua vez, são táticas, uma vez que os indicadores derivados delas são usados por um gerente de projeto e uma equipe de software para adaptar o fluxo de trabalho do projeto e as atividades técnicas e descobrir áreas problemáticas antes que elas se tornem críticas. No entanto, os autores concordam que tais métricas são responsáveis por influenciar a tomada de decisão dos gestores.

Nesta pesquisa, destacamos dois dados para o estabelecimento da métrica a ser utilizada. Visto que o objetivo era a melhoria do processo de desenvolvimento, quais atributos do processo poderiam ser medidos para verificar sua qualidade? Assim, optou-se pelo esforço e complexidade no desenvolvimento das issues. Os dados foram escolhidos uma vez que são facilmente calculáveis e entendíveis e são repetíveis independente do observador, além de poder ser expressa em uma unidade.

Para fins de medição, devemos definir que complexidade compreende: 1) o nível do impacto que a issue irá causar, ou poderá causar no sistema; 2) Amplitude da adequação/atualização referente ao sistema como um todo; 3 Nível de dificuldade técnica no desenvolvimento da rotina. Já o esforço compreende o esforço / trabalho e conhecimento para execução da issue em si.

Foi estabelecida uma relação entre os dois dados para gerar uma unidade que auxiliasse na medição dos dados.

Tabela 4 – Relação entre esforço e complexidade

	Esforço				
Complexidade	**5**	**4**	**3**	**2**	**1**
1 – Pequena	25	20	15	10	5
2 – Média	50	40	30	20	10
3 – Grande	75	60	45	30	15

Fonte: Elaborado pela autora

Para extração dos dados, as issues do projeto foram filtradas no software de gerenciamento do projeto – JIRA – e exportadas como arquivo xls onde puderam ser tabulados.

Dentro desse contexto, é padronizado e de conhecimento de todos os gestores as definições abaixo relativas a complexidade e esforço:

Tabela 5 – Descrição do Nível de Complexidade

Nível	Descrição
Grande	Rotinas ou adequações que envolvem manipulação de partes importantes do sistema e tem grande influência no restante dos processos do mesmo, ou cuja execução de codificação é de dificuldade alta, com necessidade de pesquisa para implementação no sistema.
Média	Rotinas ou adequações que envolvem a manipulação de partes importantes do sistema, mas referentes a processos específicos do sistema. Ou cuja execução de codificação de dificuldade alta, mas que já possuem exemplos no próprio sistema. Execução de codificação com necessidade de pesquisas, mas de dificuldade pequena / média.
Pequena	Rotinas ou adequações simples que manipulam partes específicas do sistema, como cadastros, procuras, relatórios. Ou cuja execução é rotineira, de codificação simples.

Fonte: Documentação da empresa

Tabela 6 – Descrição do Nível de Esforço

Nível		Descrição
5	New Feature	Rotinas com várias funcionalidades que envolvem outras partes do sistema ou necessidade de interação externa (com outros sistemas), com manipulação de várias tabelas e com tempo de execução de mais de 3 dias.
	Improvement	Reformulação da rotina como um todo, envolvendo mais de uma rotina, com tempo de execução de mais de 2 dias.
4	New Feature	Rotinas de Lançamento com funcionalidades básicas e intermediárias que envolvem outras partes ou rotinas do sistema, com manipulação de várias tabelas do sistema, com tempo de execução de mais de 2 dias.
	Improvement/Bugs	Envolvem muita codificação em uma ou mais rotinas e considerável análise para melhoria da funcionalidade e resolução de problemas, com tempo de execução de mais de 2 dias.
3	New Feature	Rotinas de cadastro com vários campos, com funcionalidades básicas e intermediárias/ avançadas, procura com filtragem por mais de um campo. Relatórios com vários filtros, que envolvem várias tabelas e apresentam estatísticas ou apresentam informações gerenciais. Lançamentos com funcionalidades básicas e intermediárias, com impressão e manipulação de mais de uma tabela, com tempo de execução de mais de 1 dia.
	Improvement/Bugs	Envolvem a codificação considerável em uma ou mais

		rotinas e certa análise para melhoria da funcionalidade ou resolução do problema, com tempo de execução de mais de 1 dia.
2	New Feature	Rotinas de cadastros com mais de 5 campos, com funcionalidades básicas (inclusão, alteração e exclusão), além de tela de procura com filtragem por mais de um campo. Relatórios ou relações com existência de 3 ou mais filtros, envolvendo mais de 3 tabelas. Lançamentos com funcionalidades básicas, sem necessidade de impressão ou com impressão simples que manipulam de 1 a 2 tabelas do sistema.
	Improvement/Bugs	Envolvem codificação de parte da rotina e certa análise para melhoria da funcionalidade ou resolução do problema, com tempo de execução de mais de 1 dia.
1	New Feature	Rotinas de cadastros com poucos campos (5 no máximo), com funcionalidades básicas (inclusão, alteração e exclusão), além da tela de procura simples. Relatórios do tipo listagem com existência de 1 ou 2 filtros, envolvendo entre 1 ou 2 tabelas.
	Improvement/Bugs	Envolvem pouca codificação e análise para melhoria da funcionalidade ou resolução do problema, com tempo de execução de no máximo 1 dia.

Fonte: Documentação da empresa

Para fins didáticos, ao invés de evidenciar os nomes das rotinas do sistema, elas foram identificadas como letras para tabulação dos dados. No entanto, podem ser identificadas também na relação abaixo:

A – Lançamento de Certificado de Registro de Fretamento
B - Lançamento de Vistoria de Fretamento
C - Cadastro de Taxista
D – Cadastro de Mototaxista
E – Ficha Cadastral de Mototaxista
F – Ficha Cadastral de Taxista
G – Procura de Veículo de Fretamento
H – Consulta de Taxista
I – Consulta de Lista de Espera de Mototaxista
J – Lançamento de Ocorrência
K – Relatório de Taxista
L - Cadastro da empresa de Fretamento
M - Cadastro de empresa para Taxista

AMOSTRAS

Buscando obter uma representação proporcional, utilizamos a amostragem aleatória estratificada. Assim: primeiramente foram identificados os subgrupos (estratos) significativos: neste caso, cada tipo de issue (bug, new feature, improvement e task). Em seguida, foi atribuído um peso relativo a cada um desses estratos de acordo com sua representação (%) na amostra e por fim, foi utilizado um procedimento de amostragem aleatória simples para escolher os sujeitos de cada estrato a integrar a amostra.

Abaixo a representação formal de cada estrato da nossa população de issues.

Tabela 7 – Amostra de issues por estrato

Tipo	População	Porcentagem (%)	Amostra
Bug	45	45	20
Improvement	29	29	8
New Feature	25	25	6
Task	2	1	1

Fonte: Elaborado pela autora

RESULTADOS

As tabelas abaixo representam os dados que foram tabulados.

Com relação ao tipo Bug, foi predominante a ocorrência de problemas principalmente com relação as rotinas B e C, que correspondem ao Lançamento de Vistoria de Fretamento e ao Cadastro de Taxista.

Tabela 8 – Relação de Rotinas vs. Issues do Tipo Bug

	Peso	Quantidade de Sprints	Rotina Relacionada
1	30	1	A
2	5	3	B
3	40	2	C
4	5	2	D
5	40	1	E; F
6	30	1	B
7	30	1	B
8	5	2	G
9	10	7	F; B
10	30	1	H
11	20	1	E; F
12	15	1	C; D
13	20	1	C
14	30	1	C
15	60	1	I
16	40	1	C
17	30	1	B
18	40	1	B
19	30	1	J
20	15	1	D

Fonte: Elaborado pela autora

As amostras de *improvement* analisadas trouxeram resultados bem balanceados se desconsiderarmos os pesos. As rotinas A, C, D, E e F, ou seja, Lançamento de Certificado de Registro de Fretamento, Cadastro de Taxista, Cadastro de Mototaxista, Ficha Cadastral de Mototaxista e Ficha Cadastral de Taxista tiveram as mesmas ocorrências.

Tabela 9 -Relação de Rotinas vs. Issues do Tipo Improvement

	Peso	**Quantidade de Sprints**	**Rotina Relacionada**
1	10	1	A
2	5	3	A
3	15	3	C; D
4	10	1	C; D
5	5	3	E
6	5	1	F
7	10	2	K
8	10	1	E; F

Fonte: Elaborado pela autora

Quanto as issues *New Features*, também houve certo balanceamento, com ocorrência similar das rotinas B (Lançamento de Vistoria de Fretamento) e M (Cadastro de empresa para Taxista).

Tabela 10 – Relação de Rotinas vs. Issues do Tipo New Feature

	Peso	Quantidade de Sprints	Rotina Relacionada
1	15	4	B
2	15	3	A
3	15	3	L
4	75	1	M
5	45	1	M
6	60	1	B

Fonte: Elaborado pela autora

Por sua vez, a rotina do tipo Task sorteada referia-se a análise de quatro rotinas diferentes: Lançamento de Certificado de Registro de Fretamento, Lançamento de Vistoria de Fretamento, Procura de Veículo de Fretamento e Cadastro da empresa de Fretamento.

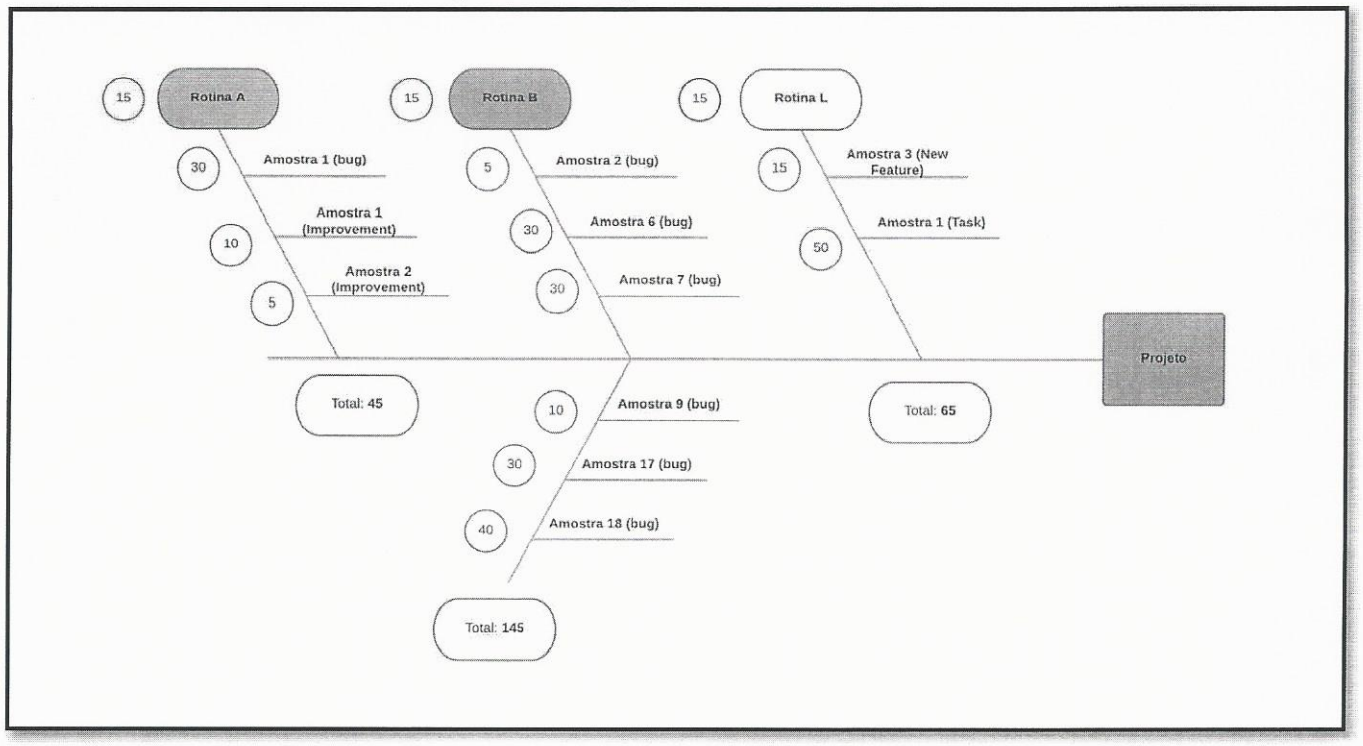
Tabela 11 – Relação de Rotinas vs. Issues do Tipo Task

	Peso	Quantidade de Sprints	Rotina Relacionada
1	50	5	A; B; G; L

Fonte: Elaborado pela autora

Foram identificadas maiores ocorrências em 23 rotinas do sistema. Visto que nossa hipótese sugeria que era possível identificar as rotinas que apresentavam mais problemas, podemos visualizar abaixo o relacionamento de algumas rotinas que foram *new features* – ou seja, novas funcionalidades do sistema – e que posteriormente apresentaram diversas solicitações:

Figura 4 – Diagrama de Causa e Efeito



Fonte: Elaborado pela autora

Foram destacadas três rotinas new feature, são elas o Lançamento de Certificado de Registro de Fretamento, Lançamento de Vistoria de Fretamento e Cadastro da empresa de Fretamento. Cada uma delas teve, segundo a relação entre esforço e complexidade estabelecida, o mesmo peso: 15. Ao longo do processo do projeto, foram lançadas outras issues relacionadas a essas rotinas.

Quanto a rotina A - Lançamento de Certificado de Registro de Fretamento foi lançado 1 (um) bug e 2 (dois) improvements. Para a rotina B - Lançamento de Vistoria de Fretamento foram abertas 6 (seis) issues do tipo bug. E por fim, para a rotina L - Cadastro da empresa de Fretamento, foi lançada uma issue task e uma issue new feature.

Observando o diagrama, nota-se que se no desenvolvimento da rotina o peso foi de 15, se considerarmos todas as issues derivadas da rotina, obtemos valores superiores ao estimado inicialmente para o processo de desenvolvimento. Apesar do processo estar padronizado e do software ter certo nível de maturidade, não puderam se antecipar e prever diversas situações que resultaram em novas issues e consequentemente, em retrabalho no ciclo de desenvolvimento.

CONSIDERAÇÕES FINAIS

Os resultados desta pesquisa serviram para reafirmar a importância do gerenciamento e planejamento para a boa execução dos projetos. Nos propomos a avaliar o processo de desenvolvimento do software através da aplicação dos conceitos apreendidos e ao longo da pesquisa pudemos entender o comportamento e funcionamento do sistema e dos seus processos.

A pesquisa pretendeu abordar o processo de desenvolvimento de um software de determinada empresa, cujas especificidades da estrutura organizacional foram consideradas. A forma de utilizar a ferramenta para gerenciamento do projeto, os padrões quanto aos dados inseridos e as particularidades do processo de desenvolvimento tornaram-se fatores ímpares. O objetivo não era mudar a estrutura organizacional nem o processo de desenvolvimento do software mas sim, contribuir com base em dados científicos e empíricos na melhoria do processo uma vez que identificou-se que um dos principais problemas do processo encontra-se na análise. Esforço e complexidade, uma vez subestimados, geraram diversas outras solicitações, conforme dados da seção anterior.

Dessa forma, acreditamos que uma possível contribuição da pesquisa é a reflexão sobre as práticas adotadas e modificação da abordagem técnica para melhorar a qualidade do produto entregue ao cliente.

REFERÊNCIAS

BRASIL é o 7º no mundo em investimentos em TI, com US$ 60 bilhões em 2014. Disponível em <http://corporate.canaltech.com.br/noticia/mercado/brasil-e-o-7o-no-mundo-em-investimentos-em-ti-com-us-60-bilhoes-em-2014-40965> Acesso em: 08 maio 2016.

BECK, K. et al. Manifesto para o desenvolvimento ágil de software. c2001. Disponível em:< >. Acesso em: 20 maio 2016.

CARVALHO, Bernardo Vasconcelos de; MELLO, Carlos Henrique Pereira. Aplicação do método ágil scrum no desenvolvimento de produtos de software em uma pequena empresa de base tecnológica. Gest. Prod., São Carlos , v. 19, n. 3, p. 557-573, 2012 . Disponível em: < http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0104-530X2012000300009&lng=en&nrm=iso >. Acesso em: 16 Maio de 2016. http://dx.doi.org/10.1590/S0104-530X2012000300009.

FISHER, J; KONING, D; LUDWIGSEN, A.P. Utilizing Atlassian Jira For Large-Scale Software Development Management. Presented at: ICALEPCS 2013, San Francisco, CA, United States, Oct 06 - Oct 11, 2013.

GIL, A.C. Como Elaborar Projetos de Pesquisa - 4. ed. - São Paulo: Atlas, 2002

GUERRA, Ana Cervigni Guerra; COLOMBO, Regina Maria Thienne. Tecnologia da Informação: Qualidade de Produto de Software. Brasília: MCT/SEPIN, 2009. 429 p.

Mercado Brasileiro de Software: panorama e tendências, 2015 = Brazilian Software Market: scenario and trends, 2015 [versão para o inglês: Anselmo Gentile] - 1ª. ed. - São Paulo: ABES - Associação Brasileira das Empresas de Software, 2015.

MITTAL, N. The Burn-Down Chart: An Effective Planning and Tracking Tool. Disponível em: < https://www.scrumalliance.org/community/articles/2013/august/burn-down-chart-%E2%80%93-an-effective-planning-and-tracki#sthash.RxfLJQs8.dpuf> Acesso em: 30 maio 2016.

PRESSMAN, Roger S. Engenharia de software: uma abordagem profissional. 7. ed. Porto Alegre: McGraw Hill, 2011. 771 p

SOMMERVILLE, Ian. *Engenharia de Software* . 6ª ed. São Paulo: Addison Wesley. *2003*.

_____, I. *Software Engineering*. Addison Wesley. *2011*

SCHWABER, K.; SUTHERLAND, J. The Scrum Guide.The Definitive Guide to Scrum: The Rules of the Game. Julho, 2013.

LISTA DAS ATIVIDADES PREVISTAS REALIZADAS E NÃO-REALIZADAS

23º Simpósio Internacional de Iniciação Científica e Tecnológica da USP. A utilização de ferramentas open source na melhoria do processo de qualidade de software. 2015.

V Congresso de Pesquisa Científica: Inovação, Sustentabilidade, ética e cidadania. Univem, FAJOPA, Fatec e FAMEMA. Melhoria contínua no processo de qualidade de software. 2015.