Análise das atividades de uma equipe ágil em uma *sprint* para integrar qualidade no Desenvolvimento de Software

Thiago Bessa[1], Marcela Pessoa[1], Luana Lobão[2]

[1]Fundação Centro de Análise, Pesquisa e Inovação Tecnológica (FUCAPI)
Caixa Postal 7200 – 69075-351 – Manaus – AM – Brasil

[2]Instituto de Desenvolvimento Tecnológico – (INDT)
Caixa Postal 7200 – 69048-660 – Manaus – AM – Brasil

Abstract: *Testing is an important activity in quality assurance of software products. However, there is still a huge shortage of qualified professionals able to work with software testing. However, not all companies are able to practice testing activities correctly. Therefore, the purpose of this project is to analyze the experience of a development team and describe their testing activities to ensure product quality and finally develop an empirical study of the main activities in a sprint.*

Resumo: *Teste é uma atividade importante na garantia da qualidade de produtos de software. No entanto, ainda há uma grande carência por profissionais qualificados e aptos para trabalhar com teste de software. Todavia, nem todas as empresas, conseguem praticar atividades de testes corretamente. Sendo assim, o intuito deste projeto é analisar a experiência de um time de desenvolvimento e descrever suas atividades de testes para garantir qualidade no produto e finalmente desenvolvendo um estudo empírico das principais atividades em uma sprint.*

1. Introdução

O mercado de Tecnologia da Informação (hardwares, softwares e serviços) no Brasil cresceu 4,5%, de acordo com o estudo anual da ABES (Associação Brasileira das Empresas de Software), realizado em conjunto com a consultoria IDC (2017). Por conta deste crescimento, o processo de teste tem sido fundamental para gerar qualidade e confiabilidade ao ciclo de vida do software, Glenford Myers (1979) afirmou que "... as considerações mais importantes no teste de software são questões de economia e psicologia humana". Estudos estimam que mais de 50% do custo de desenvolvimento de software é dedicado ao teste [B. B. 1990].

É oportuno lembrar que cada projeto apresenta características diferentes, que dependem do software, do contexto ao qual esse software irá atender, da tecnologia utilizada para o seu desenvolvimento e de outros aspectos. Assim, a escolha adequada dos tipos de testes que serão implantados torna-se um fator importante. Para que a empresa possa oferecer um produto de qualidade para seus clientes é necessário estudar e adquirir conhecimento na área para aplicar técnicas de teste de software e usar ferramentas para auxiliar no desenvolvimento.

Com essa necessidade de gerar conhecimento para os futuros profissionais surgiram várias comissões de teste e uma delas foi na Alemanha, que começou a dar suporte a um esquema de qualificação ao testador em 2002. No Brasil a ABRAMTI - Associação Brasileira de Melhoria em TI - conseguiu em 2006 o reconhecimento para a criação o BSTQB (*Brazilian Software Testing Qualification Board*). É onde se insere a certificação Certificado

Testador de Nível Fundamental - CTFL (*Certified Tester Foundation Leve*l). Essa é a mais comum, geralmente a primeira que os profissionais de testes de software procuram. Os certificados *Foundation* são: CTFL *Foundation Level*; CTFL *Agile Tester*; CTFL *Model Based Test*. O CTFL exige 65% de acerto nas questões para ser aprovado.

Além da importância para a área de teste, a relevância para a economia é substancial pois ajuda na geração de empregos e renda para a região que essas empresas de tecnologia se instalam. E, com a em conformidade com a informática apoia o crescimento tecnológico no país. De acordo com Ehmer, Mohd. Khan, Farmeena. (2014), os erros de software custam à economia dos EUA 0,6% do produto interno bruto e cerca de 80% dos custos de desenvolvimento de software de um projeto são gastos na identificação e correção de erros.

Neste contexto, o presente trabalho apresenta um relato de experiência em um ambiente que a maioria de um time de teste possui a certificação CTFL, que faz uso do conceito fundamental para contribuir no desenvolvimento e gerar qualidade do software. O objetivo é evidenciar o impacto gerado na qualidade de software quando os profissionais buscam o aperfeiçoamento na área de desenvolvimento e de teste. Para tanto, pretende-se: i) identificar quais são as atividades dos profissionais que estão no mercado competitivo na área de Software; ii) mostrar a demanda de software que aumenta a cada ano; iii) evidenciar que a atividade de teste pode ajudar na qualidade do software e iv) como a certificação pode ajudar nesta capacitação.

Este artigo está organizado como segue: a Seção 2 apresenta os trabalhos relacionados, na Seção 3 é realizada a fundamentação teórica, a Seção 4 apresenta a metodologia utilizada no trabalho, a Seção 5 apresenta a experimentação e a discussão dos resultados e, na Seção 6, é feita a conclusão.

2. Trabalho Correlatos

A área de teste tem despertando o interesse de pesquisadores e do mercado, uma vez que impacta diretamente na qualidade do produto, neste contexto, Daniel Botter (2016) apresenta um estudo na área de teste de software onde ele afirma que nem todas as empresas, principalmente as micro e pequenas, conseguem praticar essa atividade e, quando conseguem, praticam de forma incorreta. Isso porque essas empresas não encontram tempo para se planejar e estruturar para que haja um processo de teste adequado. Em alguns casos, empresas chegam a testar sem muito processo, somente quando existe a necessidade, executam o chamado de teste *ad-hoc*. Sendo assim, o intuito deste trabalho é mostrar que uma empresa pequena é capaz de se estruturar e ter um processo de teste e qualidade de software, sendo apta a tornar seus produtos mais competitivos e tornar-se exemplo de teste e qualidade.

Collins *et. al.* (2017) descreve um projeto para o desenvolvimento de capacitação na área de verificação e validação de software. Em virtude do crescimento do Pólo Industrial de Manaus (PIM) e dos incentivos fiscais providos pelo Governo Federal houve o aumento de empresas de desenvolvimento de software. O principal objetivo do projeto foi propiciar um ambiente para realização de um programa continuado de formação de recursos humanos. Com isso, promover a capacitação de profissionais para atuarem na indústria de software.

Já Valle, Barbosa e Maldonado (2015) realizaram um mapeamento sistemático sobre ensino de teste de software que identificaram as principais abordagens para o ensino de teste de software, bem como forma de desenvolvê-las e avaliá-las. Além disso, eles identificaram as linguagens alvo e as fases de teste de software consideradas nessas abordagens.

Caracterizaram o estado da arte referente ao ensino de teste, observando que as abordagens mais utilizadas são jogos educacionais e ensino de teste com programação.

Buscando demostrar a relevância do assunto de teste de software entre os trabalhos relacionados e a proposta deste artigo, a Tabela 1 apresenta informações e resultados de cada trabalho. A última lista descreve as características do trabalho proposto.

Tabela 1. Relevância dos Trabalhos Relacionados.

Ano	Tema	Objetivo	Resultados
2015	Um Mapeamento Sistemático Sobre Ensino de Teste de Software	Identificar o estado da arte referente ao ensino de teste de software.	Identificaram-se as principais abordagens para o ensino de teste de software, bem como a forma de desenvolvê-las e avaliá-las. Além disso, identificaram-se as linguagens alvo e as fases de teste de software consideradas nessas abordagens.
2016	Implantação De Teste De Software Em Empresa De Pequeno Porte: Um Estudo De Caso	Realiza um estudo de caso que demonstre o processo para que micro e pequenas empresas sejam capazes realizarem testes em seus softwares de forma correta.	O objetivo foi atingido, sendo possível implantar a metodologia de trabalho, com o uso da ferramenta para o gerenciamento dos testes, visualizando os processos de teste, as formas de testar, não antes efetuados, que agora passaram a ser registrados e testados corretamente.
2017	Inovação na capacitação de profissionais na área de Verificação e Validação de Software	O principal objetivo do projeto de colaboração, foi propiciar um ambiente para realização de um programa continuado de formação de recursos humanos.	Todas as iniciativas descritas neste artigo poderão ser aplicadas em outros locais do país com características similares às do Amazonas que é uma região do país em constante evolução na área de tecnologia.
2019	Análise das atividades de uma equipe ágil numa sprint para integrar qualidade no desenvolvimento de software	Identificar a necessidade da qualificação de profissionais que desejam ingressar na área de Testes de Software. Com base em case que foram aplicados após a obtenção das certificações.	Por meio das respostas obtidas podemos identificar as principais atividades no sprint de desenvolvimento para garantir um time competitivo no mercado e entregas de produtos validados e verificados.

3. Teste de Software

Teste de software é um conjunto de atividades que pode ser planejado com antecedência e executado sistematicamente [Pressman 2011, p.402]. O teste é uma parte inevitável de qualquer trabalho que envolve o desenvolvimento de um sistema de software (William Howden, Pressman 2011, p. 402), desta forma, nesta seção são apresentados os principais

conceitos sobre Teste de software, incluindo processos, atividades e tarefas, modelos de ciclo de vida de um software e técnicas de teste.

3.1. Atividades e Tarefas de Teste

Não existe um processo universal de teste de software, mas há conjuntos comuns de atividades de teste, sem elas os softwares terão menor probabilidade de atingir seus objetivos estabelecidos. Esses conjuntos de atividades de teste são um processo de teste. Um processo de teste consiste nos seguintes grupos principais de atividades: Planejamento do teste, Monitoramento e controle do teste, Análise do teste, Modelagem do teste, Implementação do teste, Execução do teste, Conclusão do teste. Segundo Maldonado *et. al.* (2007), as atividades devem ser executadas ao longo do processo de desenvolvimento do software e se concretizam em quatro fases de teste: de unidade, de integração, de sistema e de aceitação, cada uma delas com objetivos diferentes.

Por exemplo, o desenvolvimento ágil envolve pequenas iterações de projeto, construção e teste de software que acontecem de forma contínua, suportadas pelo planejamento contínuo. Assim, as atividades de teste também estão acontecendo de forma contínua e interativa dentro dessa abordagem de desenvolvimento. Mesmo em desenvolvimento sequencial, a sequência lógica de atividades escalonada envolverá sobreposição, combinação, simultaneidade ou omissão, de modo que a adaptação dessas atividades principais dentro do contexto do sistema e do projeto geralmente seja necessária.

3.2. Modelos de Ciclo De Vida de Desenvolvimento de Software

Um modelo de ciclo de vida de desenvolvimento de software descreve os tipos de atividades realizadas em cada estágio de um projeto de desenvolvimento de software e como as atividades se relacionam umas com as outras de forma lógica e cronológica. Há vários modelos de ciclo de vida de desenvolvimento de software, cada um dos quais requer abordagens diferentes para o teste [Syllabus 2018, p.30]. Os modelos mais comuns de ciclo de vida de desenvolvimento de software são: Modelos de desenvolvimento sequencial; e Modelos de desenvolvimento iterativo e incremental.

Um modelo de desenvolvimento sequencial descreve o processo de desenvolvimento de software como um fluxo sequencial e linear de atividades. Isso significa que qualquer fase do processo de desenvolvimento deve começar quando a fase anterior estiver concluída. Em teoria, não há sobreposição de fases, mas, na prática, é benéfico ter antecipadamente o *feedback* da fase seguinte [Syllabus 2018, p.31].

O desenvolvimento incremental é aquele em que o software/produto é construído e entregue por pedaços. Cada pedaço ou incremento representa um subconjunto de funcionalidades completas. O incremento pode ser pequeno ou grande, por exemplo, ele pode variar apenas de uma tela de relatórios simples, para um conjunto altamente flexível de telas de gerenciamento de dados. Cada incremento é totalmente codificado e testado, e a expectativa geral é que o trabalho tenha a conclusão mais completa possível [Bernardo 2015].

O desenvolvimento iterativo é aquele que faz progresso através de tentativas sucessivas de refinamento. Em cada iteração a equipe de desenvolvimento identifica e especifica os requisitos relevantes, cria um projeto utilizando a arquitetura escolhida como guia, implementa o projeto em componentes e verifica se esses componentes satisfazem os requisitos. Se uma iteração atinge os seus objetivos, o desenvolvimento prossegue com a

próxima iteração, caso contrário a equipa deve rever as suas decisões e tentar uma nova abordagem [Bezerra 2017, p.37]. A Figura 1 ilustra o Ciclo de vida iterativo e incremental.

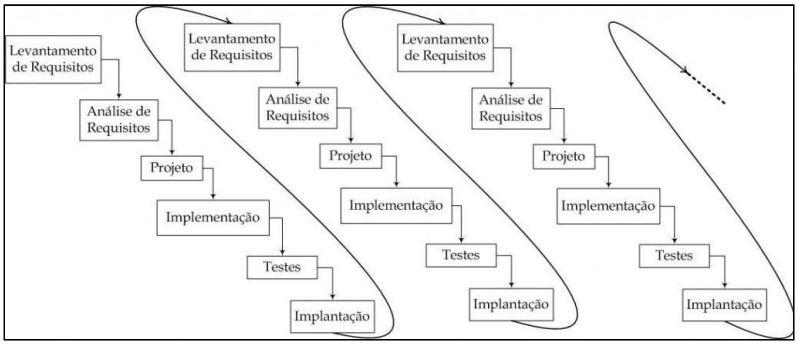

Figura 1: Diagrama do Ciclo de Vida Iterativo e Incremental

Além disso, os próprios modelos de ciclo de vida de desenvolvimento de software podem ser combinados. Por exemplo, um Modelo V pode ser usado para o desenvolvimento e teste dos sistemas de apoio e suas integrações, enquanto um modelo de desenvolvimento Ágil pode ser usado para desenvolver e testar a interface do usuário e a funcionalidade. A prototipagem pode ser usada no início de um projeto, com um modelo de desenvolvimento incremental adotado uma vez que a fase experimental esteja completa. Como estratégia para execução do processo de teste, recomenda-se a utilização do modelo "V" que é um modelo onde todas as fases de desenvolvimento podem ter uma fase de teste associada. Ele ajuda a pensar em teste o quanto antes no processo de desenvolvimento de software.

3.3. Técnicas de Teste

As técnicas de teste são classificadas como caixa-preta, caixa-branca ou baseada em experiência.

As técnicas de teste caixa-preta (também chamadas de técnicas comportamentais ou baseadas no comportamento) focaliza os requisitos funcionais do software. As técnicas de teste caixa-preta permitem derivar séries de condições de entrada que utilizarão completamente todos os requisitos funcionais para um programa. O teste caixa-preta não é uma alternativa às técnicas caixa-branca. Em vez disso, é uma abordagem complementar, com possibilidade de descobrir uma classe de erros diferente daquela obtida com métodos caixa-branca. O teste caixa-preta tenta encontrar erros nas seguintes categorias: funções incorretas ou faltando, erros de interface, erros em estruturas de dados ou acesso a bases de dados externas, erros de comportamento ou de desempenho, e erros de inicialização e término [Pressman 2011, p.439].

As técnicas de teste caixa-branca (também chamadas de técnicas estruturais ou baseadas na estrutura) é uma filosofia de projeto de casos de teste que usa a estrutura de controle descrita como parte do projeto no nível de componentes para derivar casos de teste. Usando métodos de teste caixa-branca, o engenheiro de software pode criar casos de teste que garantam que todos os caminhos independentes de um módulo foram exercitados pelo menos uma vez, exercitam todas as decisões lógicas nos seus estados verdadeiro e falso, executam todos os ciclos em seus limites e dentro de suas fronteiras operacionais, e exercitam estruturas de dados internas para assegurar a sua validade [Pressman 2011, p.431].

As técnicas de teste baseadas em experiência aproveitam a experiência de desenvolvedores, testadores e usuários para projetar, implementar e executar testes. Estas

técnicas são frequentemente combinadas com técnicas de teste caixa-preta e caixa-branca. [Syllabus 2018, p.33].

4. Metodologia

O presente estudo utilizou-se do método de análise de dados, amparado pelo procedimento de pesquisa bibliográfica, documental, descritiva, exploratória com a finalidade de apresentar um estudo de caso. Para isso, a pesquisa será baseada em estudos de autores, como por exemplo Glenford Myers, William Howden, Eduardo Bezerra, José Carlos Maldonado, Kleber Bernardo entre outros pensadores que elaboraram trabalhos pertinentes ao assunto. O trabalho foi dividido em etapas: i) Seleção dos artefatos a serem utilizados, ii) Coleta dos Dados, iii) Análise dos dados, que podem ser visualizadas na Figura 2.

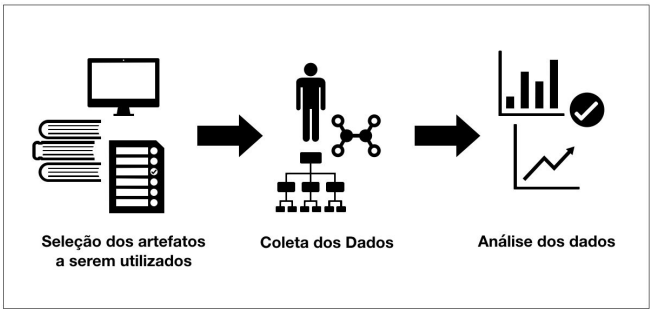

Figura 2: Etapas de trabalho para análise de dados

Na fase de Seleção dos artefatos a serem utilizados foram analisadas planilhas, formulários e outros artefatos disponíveis na literatura. Como resultado desta fase selecionou-se a planilha que está ilustrada na Figura 3, por *Sprint Planning, Divisão de Task*, Métricas das *Stores*, Planejamento do teste, Reunião de Alinhamento, *Sprint Review/Retro*, Conclusão do Teste, Análise do teste, Criação Casos de Teste, Execução do Casos de Teste, Automação, Teste de Regressão, Monitoramento e controle do teste, Teste Exploratório, Caixa Branca e Desenvolvimento.

A fase de Coleta dos Dados, está acontecendo em uma empresa de desenvolvimento de software e sendo embasados em artigos, livros, revistas especializadas, websites e documentos da empresa. Na fase de Análise de Dados, foi realizada análise sobre os dados já coletados na empresa, que são melhores descritos na Seção de resultados.

Nome:						Cargo:				
Atividades	Dia 1	Dia 2	Dia 3	Dia 4	Dia 5	Dia 6	Dia 7	Dia 8	Dia 9	Dia 10
Sprint Planning										
Divisão de Task										
Métricas das Stores										
Planejamento do teste										
Reunião de Aliamento										
Sprint Review										
Conclusão do teste										
Análise do teste										
Criação de Test Case										
Execução do teste Case										
Automação dos Test Case da Sprint Passada										
Teste de Regressão										
Monitoramento e controle do teste										
Teste Exploratório										
Teste de Caixa Branca										
Desenvolvimento - Code										

Figura 3: Planilha de atividades de uma sprint

5. Experimentos e Resultados

Em virtude da diversidade de critérios de teste existente, saber qual deles deve ser utilizado ou como utilizá-los de forma complementar a fim de obter o melhor resultado na questão de qualidade é uma questão complicada. Do ponto de vista teórico, os critérios baseados em análise de fluxo de dados têm complexidade exponencial [Maldonado 1991] o que motiva a condução de estudos empíricos para determinar o custo de aplicação desses critérios do ponto de vista prático.

Os experimentos foram realizados em um instituto de desenvolvimento de software em Manaus contou com as respostas de 11 pessoas (número máximo de pessoas disponíveis para participar), dentre elas 6 pertenciam à equipe de desenvolvedores e 5 à equipe de teste, conforme ilustra a gráfico 1. Ressaltando que todos da equipe de teste foram aprovados pelo BTSQB adquirindo a certificação CTFL.

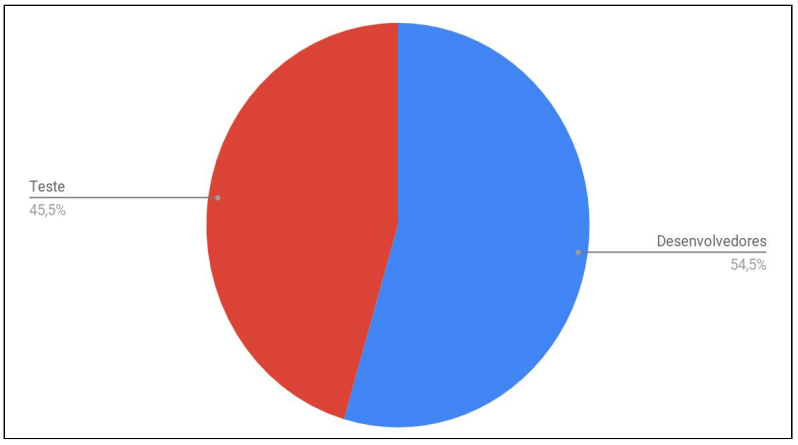

Gráfico 1: Quantidades de Atividades por dia numa sprint

As funcionalidades desenvolvidas nessa *sprint* foram testadas seguindo o teste exploratório e a técnica de caixa-branca. Os resultados foram considerados aceitos por conta que todos os critérios de aceite que foram submetidos nas métricas no início da *sprint* foram desenvolvidas pelo time. Com várias atividades podemos analisar a média de atividades por dia que um time de desenvolvimento no Gráfico 2. Possível observar no eixo Y a representação da quantidade de atividades feitas durante um dia e no eixo X o tamanho da *sprint* que contabiliza 10 dias corridos. Com essa análise de atividades por dia chegou uma média de 6 atividades distintas que o time de desenvolvimento faz durante o dia além de desenvolver requerendo outras atividades até o final da *sprint* como podemos observar o pontilhado vermelho.

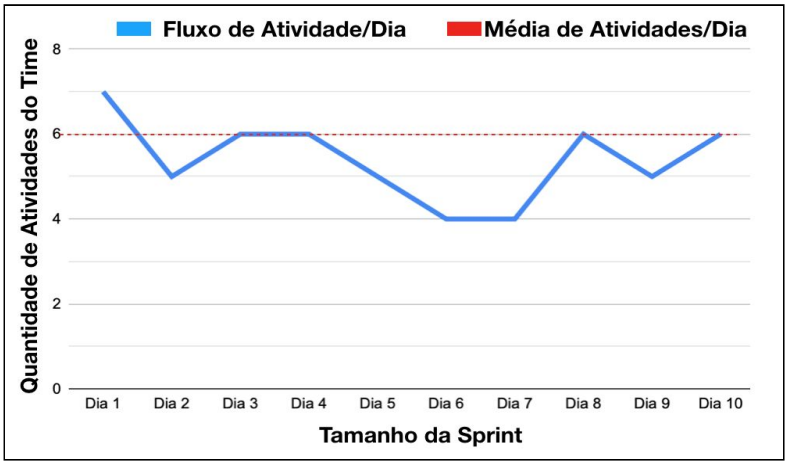

Gráfico 2: Quantidades de Atividades por dia numa sprint

Por meio das respostas obtidas foi possível identificar as principais atividades na *sprint* de desenvolvimento. Possível observar no Gráfico 3 que no eixo Y é representada a quantidade de dias que as atividade se repetem e no eixo X as atividades que ocorrem durante a *sprint. O* maior índice é o desenvolvimento do produto onde é a atividade que abrange quase toda a *sprint* por causa que estamos analisando um time com o modelo de desenvolvimento sequencial, iterativo e incremental que no final da sprint precisa entregar uma nova parte do software. Em seguida o segundo e terceiro maiores índices estão envolvidos com testes que são o de caixa-branca e teste exploratório para assegurar a qualidade do produto o qual vai ser entregue pela parte dos desenvolvedores para o cliente. Já os menores índices estão em um só dia porque são atividades que acontecem no início ou no fim requerendo só algumas horas da *sprint* como atividades de: Reunião de Alinhamento, Planejamento de teste e Métricas das histórias acontecem no primeiro de dia e somente já a Conclusão dos Testes e Reunião da Review/Retrô acontece na última dia da *sprint*.

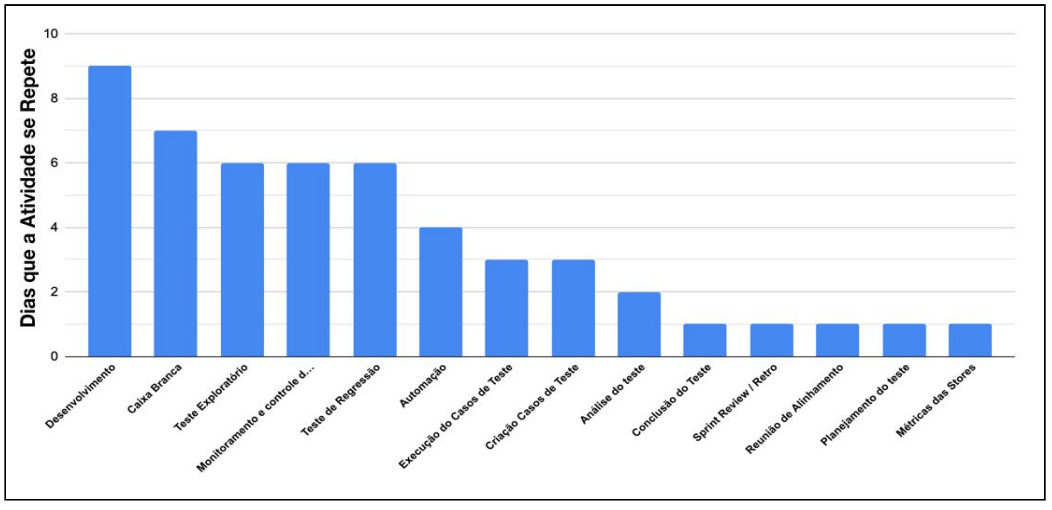

Gráfico 3. Atividades repetidas durante uma sprint de time ágil

6. Conclusão

Por meio do estudo empírico realizado neste trabalho, observou-se que há diversas atividades desenvolvidas, de forma incremental, pelo time para entregar um produto funcional. Para a

realização dessa análise foi selecionado um time com 11 pessoas. A partir dos estudos selecionados identificaram-se 14 atividades diferentes para auxiliar na qualidade da entrega no desenvolvimento de software, dentre elas: desenvolvimento de algoritmos, teste de caixa-branca feita pelos desenvolvedores e teste de caixa-preta que é o teste exploratório feito pelos testadores, as mais utilizadas.

Aproximadamente são 6 atividades diárias realizadas pelo time de desenvolvimento que abrange os desenvolvedores e testadores. Neste cenário, percebe-se a importância do ambientes envolver desenvolvedores e testadores na entrega de um único produto validado e verificado para gerar valor pro cliente. Essas atividades são compostas do Scrum e do Movimento Ágil para garantir um time competitivo no mercado. Por isso o time precisa ser organizado e auto gerenciável para cumprir as demandas de cada *sprint* entregar um produto funcional. Sendo assim, como trabalhos futuros pretende-se desenvolver e analisar outras atividades que abordam um conjunto de melhorias no processo de implementação de software. Neste cenário o estudo empírico desenvolvido contribuiu para identificar as principais atividades para o desenvolvimento de software. Por fim, para a validação seria realizada uma avaliação empírica por meio de experimentos controlados.

O estudo realizado apresentou limitações importantes quanto à seus resultados e amostra. Os resultados para este estudo foram dentro do objetivo portanto com as características do projeto escolhido influenciou a obtenção dos resultados por sua vez analisar uma equipe que tem experiência e certificados podendo abranger outras equipes que estão iniciando sua profissão e sem certificações. A definição da amostra também pode ser considerada um fator limitante tendo em vista o fato dela não ser aleatória. Outra limitação importante se refere ao tamanho da amostra, que ao se apresentar em número reduzido, permite considerar os resultados encontrados apenas para a população em questão.

8. Referências

B. B. Beizer, Software Testing Techniques, 2nd ed. Van Nostrand Reinhold Co. New York, NY, USA, 1990.

BSTQB. (2018), http://www.bstqb.org.br/, Abril.

BERNARDO, Kleber. (2015). Iterativo e incremental: Suas definições, https://www.culturaagil.com.br/iterativo-e-incremental-suas-definicoes, Agosto.

BEZERRA, Eduardo. Princípio de Análise e Projeto de Sistemas com UML. Rio de Janeiro : Eisevier, 2007.

COLLINS, Eliane. MACEDO, Gisele. LOBÃO, Luana. NETO, Arilo. (2017). Inovação na capacitação de profissionais na área de Verificação e Validação de Software.

CORREIA, Tiago. Interpretação e Validação científica em pesquisa qualitativa. Interface: Comunicação, Saúde e Educação. V.17, n. 45, p. 263-74, abr./jun. 2013.

EHMER, Mohd. KHAN, Farmeena. (2014). Importance of Software Testing in Software Development Life Cycle.

ISTQB. (2011), https://www.istqb.org/, Abril.

IDC. (2017) “Investimentos em TI no Brasil aumentam 4,5% em 2017”, http://www.abessoftware.com.br/dados-do-setor/estudo-2018--dados-2017, Junho.

MALDONADO, José Carlos; DELAMARO, Márcio Eduardo; JINO, Mario. Introdução ao Teste de software. Rio de Janeiro: Elsevier, 2007.

Revista Engenharia de Software. 18 ed. (2019). A influência na Qualidade de Software, https://www.devmedia.com.br/artigo-engenharia-de-software-18-fatores-humanos/14802, Maio.

Myers, G.J. The Art of Software Testing. Wiley, New York, 1979.

MALDONADO, J. C. *Critérios Potenciais Usos: Uma Contribuição ao Teste Estrutural de Software*. PhD thesis, DCA/FEE/UNICAMP, Campinas, SP, July 1991.

PRESSMAN, Roger S. Engenharia de software [recurso eletrônico] : uma abordagem profissional / Roger S. Pressman ; tradução Ariovaldo Griesi ; revisão técnica Reginaldo Arakaki, Julio Arakaki, Renato Manzan de Andrade. – 7. ed. – Dados eletrônicos. – Porto Alegre : AMGH, 2011.

SANTOS, Daniel. (2016). Implantação De Teste De Software Em Empresa De Pequeno Porte: Um Estudo De Caso.

SYLLABUS. (2018). Brazilian Software Testing Qualifications Board - Tradução realizada pelo WG-Traduções do BSTQB do syllabus do ISTQB, https://www.bstqb.org.br/uploads/syllabus/syllabus_ctfl_2018br.pdf, Julho.

VALLE, Pedro; BARBOSA, Ellen; MALDONADO, José C. (2015). Um Mapeamento Sistemático Sobre Ensino de Teste de Software.